DOCUMENT RESUME

ED 211 067 IR 009 887

TITLE Annual Report. July 1, 1980-June 30, 1981. Department of Computer and Information Science, The Ohio State University.

INSTITUTION Ohio State Univ., Columbus. Dept. of Computer and Information Science.

PUB DATE 30 Jun 81

NOTE 81p.; For a related document, see ED 196 451.

AVAILABLE FROM Computer and Information Science Research Center, Ohio State University, 2036 Neil Avenue Mall, Columbus, OH 43210 (complimentary copies available as long as the supply lasts).

EDRS PRICE MF01/PC04 Plus Postage.

DESCRIPTORS Annual Reports; College Faculty; *Computer Science Education; Doctoral Degrees; Higher Education; *Information Science; *Research Projects

ABSTRACT

This annual report provides information on instructional programs at both the undergraduate and graduate levels and the computing facilities of the Department of Computer and Information Science, and briefly describes the interactions of the department within the university and within the professional community. A list of students awarded doctoral degrees in 1980-81 includes the dissertation title and advisor's name; abstracts of the dissertations are presented in a separate section. Four research papers summarize selected on-going research by various faculty members in the department, and a summary of grants and contracts awarded for the 1980-81 year is given. Appendices include statistical data on the current status and history of the department, a list of courses offered with number and title, information on department faculty and staff members, a list of seminars conducted during the year, and lists of publications, recent technical reports, and activities of faculty and staff. (IIS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED211067

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

ANNUAL REPORT

JULY 1, 1980 - JUNE 30, 1981

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

THE OHIO STATE UNIVERSITY

COLUMBUS, OHIO 43210

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

C. Taylor

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR009887

FOREWORD

This publication contains the annual report of the Department of Computer and Information Science and a summary of the research which has been carried on during the 1980-81 academic year. This research has been supported in part by grants from governmental agencies and industry, as well as by The Ohio State University. Also included in this report is a list (Appendix I) of those students who are in the final stages of research leading to the Ph.D. degree.

The Department of Computer and Information Science is a separate academic unit located administratively in the College of Engineering, operating in part as an interdisciplinary program with the cooperation of many other departments and colleges throughout the University. Under the Department is the Computer and Information Science Research Center which is the publishing outlet for a technical report series. Research of the faculty and graduate students in the Department of Computer and Information Science is reported periodically in this series. A bibliography of recent technical reports published by the Center is included in this publication as Appendix F. Copies of some of these reports are still available on a complimentary basis from the Computer and Information Science Research Center, The Ohio State University, 2036 Neil Avenue Mall, Columbus, Ohio 43210. Titles with PB or AD numbers may be obtained from the National Technical Information Service, The U. S. Department of Commerce, 5285 Port Royal Road, Springfield, Virginia, 22161, in paper copy, magnetic tape, or in microfiche. Titles with ED numbers may be obtained from the ERIC Document Reproduction Service, P. O. Box 190, Arlington, Virginia, 22210. There is a nominal charge for their service.

Lee J. White, Chairman
Department of Computer &
Information Science
September 1981

TABLE OF CONTENTS

FOREWORD ii

I. THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE 1

INSTRUCTIONAL PROGRAMS 1
Undergraduate Programs 1
Graduate Programs 2
Course Offerings 5
Faculty 5
COMPUTING FACILITIES 5
INTERACTION WITHIN THE UNIVERSITY 7
INTERACTION WITHIN THE COMPUTER AND INFORMATION SCIENCE COMMUNITY 7
DOCTOR OF PHILOSOPHY DEGREE 8

II. RESEARCH IN COMPUTER AND INFORMATION SCIENCE 9

SELECTED RESEARCH PAPERS 9
DESPERANTO: A Distributed Processing System 9
Development of a High Level Specification Language to Enforce Resource Sharing 15
Customizable Workstation Environment 22
Current Research in Computer Graphics 28
ABSTRACTS OF PH.D. DISSERTATIONS 1980-81 38
RESEARCH AND DEVELOPMENT AWARDS 44
Equipment Grant 44
Graduate Training Grant 44
Research Grants 45

III. APPENDICES

A CURRENT STATUS AND CAPSULE HISTORY OF DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE 49

B COMPUTER AND INFORMATION SCIENCE COURSE LISTING BY NUMBER AND TITLE 50

C COMPUTER AND INFORMATION SCIENCE FACULTY 53

D COMPUTER AND INFORMATION SCIENCE SEMINAR SERIES 59

E PUBLICATIONS OF THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE STAFF 62

F RECENT TECHNICAL REPORTS 65

G ACTIVITIES OF THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE STAFF 67

H DOCTORATES AWARDED 71

I STUDENTS IN THE FINAL STAGES OF RESEARCH LEADING TO THE PH.D. DEGREE 76

I. THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

The Department of Computer and Information Science is a separate academic unit located administratively in the College of Engineering. In addition, the Department works closely with a number of other departments and colleges throughout the University. Degrees may be obtained in the Colleges of Mathematical and Physical Sciences and in the College of Administrative Science in addition to the College of Engineering. The Department has an enrollment of about 200 graduate students and 800 undergraduate majors.

The program at The Ohio State University emphasizes education, research, and the professional practice and application of computer and information science. The Department offers undergraduate and graduate degrees through the Ph.D. The research activities are a central part of the program, and are supported by many sponsoring agencies, including the Army Research Office, the Air Force Office of Scientific Research, the Office of Naval Research, and the National Science Foundation. These research activities will be described in this report.

INSTRUCTIONAL PROGRAMS

The program of the Department of Computer and Information Science is broad and extensive. The number of students enrolled in all courses was 10,659. A total of 125 students received baccalaureate degrees, 80 students received the M. S. degree, and 6 students received the Ph.D. degree. The number of applications for graduate study during this period was 525. Seventy graduate students received support from the department. There was a total of 23 full-time faculty and 7 part-time faculty. For additional statistics, see Appendix A.

Undergraduate Programs

A Bachelor of Science in Computer and Information Science (B.S.C.I.S.) is offered through the College of Engineering. This program offers the student a general education in engineering, physical sciences, and mathematics, along with intensive study in computer and information science. Substantial revisions to the program have been made this year. Courses in computer architecture and operating systems have been added to the core curriculum of programming, file design, and numerical methods. In addition, the major areas of specialization have been refined into a series of options which replaces the previous technical elective plan. Four well-defined options are now available: Software Systems, design and implementation of software with an emphasis on the problems of software engineering; Hardware-Software Systems, interaction between hardware and software, especially in embedded systems; Information Systems, design and implementa-

tion of information systems; Biomedical Computing (Pre-Med in Engineering), applications of computers in biomedical computing. As an alternative to these options, a student and his faculty adviser may tailor a program to suit his special interests and submit it for approval under the Individualized Option Plan. Both a Bachelor of Science (B.S.) and a Bachelor of Arts (B.A.) with a major in computer and information science are offered through the College of Mathematical and Physical Sciences, which is one of the coalition of colleges which compose the Colleges of the Arts and Sciences. These programs combine a broad liberal arts background with specialized study in computer and information science. The B.S. program has a more scientific and technical orientation; the B.A. program permits the student to combine the study of computer science with work in a related field of potential computer application

A Bachelor of Science in Business Administration (B.S.B.A.) with a major in computer science is offered through the College of Administrative Science. This program is designed to educate students in the technical aspects of computer and information science so that they can effectively use the information processing and problem solving capabilities of computers in business organization and management. These students follow a special sequence of introductory courses which emphasize languages and techniques commonly employed in commercial computing.

Graduate Programs

The Department offers programs leading to both master's and Ph.D. degrees.

General Requirements

Students should be able to complete a master's degree in one year of full time study (four quarters). A student will normally take a total of four years to complete a Ph.D. program.

Each student is expected to take a course of study corresponding to one of the following nine options.

OPTION I	for the student desiring a theoretical foundation in computer and information science.
OPTION II	for the student specializing in information systems.
OPTION III	for the student specializing in computer systems.
OPTION IV	for the student specializing in numerical analysis.
OPTION V	for the student specializing in operations research.

OPTION VI	for the student specializing in biomedical information processing.
OPTION VII	for the student specializing in administrative science.
OPTION VIII	for the student specializing in mathematics.
OPTION IX	for students specializing in computer hardware and software who have appropriate undergraduate background.

Each of these options provides a background in several aspects of computer and information science, as well as additional mathematical sophistication appropriate to the student's interest. Each of the options may lead to the doctoral program in computer and information science or to the master's degree.

The Master of Science degree may be considered to be either a terminal degree leading to the professional practice application of one phase or another of computer and information science or it may be considered as the first step towards the Ph.D. degree.

The Core Program

All courses of study require the completion of a core program in computer and information science which consists of the following.

Course			Credit
CIS 707	-	Mathematical Foundations of Computer and Information Science II	3
CIS 755	-	Programming Languages	3
CIS 760	-	Operating Systems	3
CIS 775	-	Computer Architecture	3
CIS 780	-	Analysis of Algorithms	3
CIS 885	-	Seminar on Research Topics in Computer and Information Science	1
CIS 889	-	Advanced Seminar in Computer and Information Science	2
TOTAL CREDIT HOURS IN CORE (Changed 1981)			18

Master of Science Program

Suggested courses of study, which complete each of the options, consist of additional electives in computer and information science, mathematics and cognate areas. The minimum number of credit hours required for the master's degree is 48 credits for Plan A (with thesis) or 53 credits for Plan B (without

thesis). Certain options of the M.S. program may require more than this minimum of credit hours. Every candidate on Plan A is required to write an M.S. thesis and successfully defend that thesis in a final examination while those on Plan B must demonstrate their mastery of the fundamentals of computer and information science by passing the M.S. Comprehensive Examination. However, a student who has passed the Ph.D. General Examination is eligible to receive the master's degree without having to satisfy either of the above requirements, and students planning to study for the Ph.D. are encouraged to obtain the M.S. degree in this manner. In the Comprehensive Examination, the student will be examined on the content of the core courses and one of the nine M.S. options.

Joint Master of Science Program with Mathematics

A special program is available so that a student may receive two master's degrees, one in mathematics and one in computer and information science, after completing 76 quarter hours of course work. Further information about the joint program may be obtained by request.

Doctoral Program

The award of the Ph.D. degree implies that the recipient achieved a mastery of a subject which allows him to work in a particular field in a creative capacity and to stimulate others working in this area. The Qualifying, General, and Final Examinations, taken by the student at various stages of his doctoral studies enable the faculty to ensure that only students of outstanding scholastic ability continue on to receive the doctoral degree.

The doctoral program emphasizes research and the Department encourages prospective Ph.D. candidates to involve themselves in research under the supervision of a faculty member at the earliest possible opportunity.

A major area is generally chosen from the active areas of faculty research. It will normally be the area in which the student expects to perform dissertation research. In fact, the General Examination is designed, among other things, to ensure the student's readiness to undertake dissertation research.

A cognate area may be elected for the minor areas of specialization. A cognate field is defined as a field supporting or closely related to the Departmental fields and is ordinarily specified by an integrated program of study in other departments of the University. Note, however, that the Ph.D. program can be very flexible and suited to the interests of each individual student.

The General Examination is usually taken in about the 9th quarter of residence and consists of appropriate written and oral portions. The second stage is completed and the student admitted to candidacy when he has received credit for a total of at least

90 quarter hours of graduate work and passed the General Examination.

The third stage, after admission to candidacy, is devoted primarily to research and seminars, the preparation of the dissertation, and the Final Examination. The Final Examination is oral and deals intensively with the candidate's field of specialization.

The Department does not have a foreign language requirement for the M.S. or Ph.D. degree.

Course Offerings

Currently there are about 90 courses (each one quarter in length) offered by the Department, 30 of which are largely undergraduate with the remainder being primarily graduate courses. In addition to these courses there are over two hundred courses offered by a variety of departments of the University which are of interest to our graduate students who are encouraged to take these courses. See Appendix B for a listing of courses by number and title.

Faculty

The Department of Computer and Information Science has a full time faculty with a wide range of backgrounds and experience. The faculty is supplemented by staff who have joint appointments with other departments; by staff from other departments or by visiting faculty who teach courses primarily for Computer and Information Science students; by adjunct staff people who are employed in off-campus organizations who teach in the Department of Computer and Information Science (See Appendix C).

COMPUTER FACILITIES

Computer Centers

There are three computer centers at The Ohio State University. They are: Instruction and Research Computer Center (IRCC), the Hospital Computer Center, and the University Systems Computer Center.

The primary computing facilities used by CIS students and faculty are those operated and supported by IRCC. These include an Amdahl 470 V/6 computer, which supports both batch and time-sharing systems in a Multiple Virtual Storage Environment (MVS). Computer facilities include nearly 200 terminals dedicated to a job entry, submittal, and retrieval system called WIDJET, which is used for most introductory programming courses and is a product of the University of Waterloo. More advanced students have access to time-sharing systems using the IBM Time-Sharing Option (TSO) and WYLBUR. A new IBM 4341 computer system using VM/CMS will offer another interactive computing alternative. These systems are tied together with a network switch, allowing a user to select the system most appropriate to his/her needs.

IRCC/CIS Computing Laboratory

The principal research resource of the Department of Computer and Information Science is the IRCC/CIS Computing Laboratory. The laboratory was

specifically designed to serve the specialized needs of problem oriented research and instruction impacting on the computer and information sciences. Consequently, it is maintained as a state-of-the-art facility for research and instructional programs in which the computing process is an object of study or is directly and actively involved as an integral element of problem formulation and solution.

The IRCC/CIS Computing Laboratory is administered and operated by the Instruction and Research Computer Center separately from the Center's main service installation. The Laboratory, thus, is maintained primarily for the Department of Computer and Information Science. This arrangement, unique among major universities, permits the Laboratory to be dedicated to research and instruction in the computer and information sciences.

The principal computer of the Laboratory is a DECsystem 20/20 time-sharing system, produced and maintained by the Digital Equipment Corporation, with the following features:

- TOPS-20 operating system
- 256K words of virtual address space for each user
- Two disk drives with a total on-line capacity of 55 million words of storage
- Tape drive capable of processing standard 1/2 inch magnetic tape at 800 or 1600 bpi densities
- A variety of CRT and hardcopy terminals and printers
- Direct-wired high-speed lines and remote dial-up lines
- High-speed paper tape reader/punch unit
- Several graphics peripherals, including several TEKTRONIX display devices and a remotely coupled AG-60 Plasma Graphics Panel.

DEC-supported compilers for the DECsystem 20/20 include FORTRAN, BASIC-PLUS, and ALGOL. DEC also provides a variety of software packages for program development and debugging, and for test editing and production. Locally-supported languages include PASCAL, LISP, SNOBOL, and BLISS.

Distributed Systems Computing Laboratory

A seven-node fault-tolerant, double-loop network is presently under construction by the CIS Department and constitutes an important research facility for faculty and graduate students. The network configuration includes a host computer node consisting of the DECsystem 20/20; each of the other six nodes will consist of a DEC 11/23 microcomputer; since each 11/23 unit will be complemented by 128K of MOS memory, a UNIX operating system, a CRT terminal, and a dual floppy disk, it can be operated stand-alone as well as a network resource to be shared. Each node of this network will be equipped with a loop-interface unit, presently under design and development by the Department.

Research in networking techniques, distributed processing hardware and software configurations, parallel processing algorithm development, and distributed data bases can be conducted on a network system available for experimental studies. Research into a number of issues of software engineering, including reliability of distributed software, computer program testing, and distributed system language development can be conducted using a

realistic and flexible computer network setting.

Database Systems Research Laboratory

The Database Systems Research Laboratory was established in 1980 in order to investigate experimentally a number of database research issues. These issues include:

1) information systems requirements of the 1990's;
2) the design and analysis of database computers to meet those requirements;
3) the use of both current and emerging technology for prototyping the design of databases and database computers which will meet these needs;
4) research into VLSI techniques for implementation of the prototyped components and database techniques.

The laboratory facilities presently include a VAX 780 and two 11/44 minicomputers, allowing for an experimental investigation of various database partitioning algorithms as retrieval tasks are assigned to these two minicomputers by the VAX processor. This equipment and laboratory are being jointly funded by Digital Equipment (DEC), the Office of Naval Research (ONR), and The Ohio State University.

INTERACTION WITHIN THE UNIVERSITY

The Department of Computer and Information Science interacts with other departments and research programs within the University because of the multidisciplinary nature of the activities encompassed in this field. A number of the academic faculty have joint appointments in other departments. Staff members of the Department of Computer and Information Science have appointments in the following departments and organizations:

a. Accounting
b. Allied Medicine
c. Art
d. Electrical Engineering
e. Engineering
f. Instruction and Research Computer Center
g. Mathematics
h. Psychology
i. University Libraries
j. University Systems Computer Center

INTERACTION WITHIN THE COMPUTER AND INFORMATION SCIENCE COMMUNITY

Columbus, Ohio, is one of the major centers for information science and for the transfer of information in the United States. A number of organizations are involved with the activities of computer and information science. This affords an opportunity for students and faculty to interact with appropriate personnel in these organizations. Some of these are:

a. Chemical Abstracts
b. Battelle Memorial Institute
c. Bell Laboratories
d. Bank One
e. Columbus and Southern Ohio Electric Company
f. Western Electric Corporation
g. Rockwell International Corp.
h. AccuRay
i. State of Ohio Department of Finance; Department of Highways
j. Columbus Board of Education
k. Ohio College Library Center

There are a large number of scientists who come to Columbus in order to visit the Department and who usually present a seminar. The lectures and seminars for the period of this report are listed in Appendix D.

Research efforts of the staff are disseminated to the professional community through several publication channels. A list of current publications of the Department staff is included as Appendix E. In addition, the Research Center issues a technical report series (see Appendix F for reports issued from 1979 to date). Our faculty attends most of the major technical meetings in this country as participants giving papers, assisting on panels, as attendees, and as officials. A list of these activities can be found in Appendix G.

DOCTOR OF PHILOSOPHY DEGREE

The Doctor of Philosophy degree was awarded to the following students during 1980-81. Abstracts of these dissertations are included on pages 38-43. See Appendix H for a complete list of doctorates awarded.

Name	Dissertation	Advisor
Chou, Chuen-Pu	System Design of the Distributed Loop Database System (DLDBS)	Liu
Li, Chung-Ming	Communicating Distributed Processes: A Programming Language Concept for Distributed Systems	Liu
Mittal, Sanjay	Design of a Distributed Medical Diagnosis and Data Base System	Chandrasekaran
Tsay, Duen-Ping	MIKE: A Network Operating System for the Distributed Double-Loop Computer Network	Liu
Wang, Pong-Sheng	Computer Architecture for Parallel Execution of High-Level Language Programs	Liu
Wu, Shyue Bin	Interconnection Design and Resource Assignment for Large Multi-Microcomputer Systems	Liu

Students who have passed the Ph.D. General Examination and are in the final stages of research leading to the Ph.D. degree are listed in Appendix I.

II. RESEARCH IN COMPUTER AND INFORMATION SCIENCE

Research Programs

The following four papers summarize selected on-going research by various faculty members in the Department. These papers are followed by abstracts of Ph.D. dissertations from 1980-81, and by a summary of grants and contracts awarded for the 1980-81 academic year. These selected papers, abstracts, and research awards give an overview of the primary research projects in the Department.

SELECTED RESEARCH PAPERS

1. DESPERANTO: A DISTRIBUTED PROCESSING SYSTEM

Faculty: S. A. Mamrak, T. S. Berk, D. W. Leinbaugh

Introduction

Many distributed computer systems are being formed by linking together already fully functioning, heterogeneous, single computer sites. These single-site computers often offer a full range of services which have been coded to run under local operating systems. There is a recognized need for a distributed software support system to allow sharing of such already existing services and to allow the development of new, distributed services.

Desperanto, an Esperanto for Distributed systems, is a comprehensive software system designed to support the sharing of resources in a local-area network consisting of heterogeneous computer sites (Ref. [7]). The Desperanto system consists of both programming environment and run time environment components that are in various stages of design. This report presents the motivation for Desperanto's design and a brief overview of the various components which comprise the Desperanto system.

The Motivation for Desperanto

Building a distributed processing support system for already functioning heterogeneous computer sites requires designing software that has to interface both with already existing applications and with already existing applications and with already existing operating systems. In most cases, both the application and the operating systems represent thousands of lines of working code and thousands of hours of human effort. A primary design goal of a well-designed distributed support system, therefore, should be to provide for installation of the system with as little change to existing resources as possible. Desperanto's design already allows us to achieve this goal for existing applications. We are currently investigating how the goal might be achieved for existing operating systems.

Distributed processing systems of various kinds have been designed and in some cases implemented [2, 3, 4, 5, 8, 9, 10, 11]. These efforts can be characterized as having taken a "bottom-up" view of a distributed support system. That is, they take the traditional operating system view

that the distributed support should consist of a uniform layer of software that provides a set of standard services to all applications above it. This uniform layer is typically implemented in a site-dependent manner, either in cooperation with a local site operating system or in place of it.

A key concept in Desperanto's design, and the one that sets it apart from other similar efforts, is that the software support has been conceptualized from the top-down rather than from the bottom-up. Desperanto views the software support needed for distributed sharing as it relates to individual applications or resources rather than as it relates to individual sites. This point of view has provided Desperanto with the flexibility to support a diverse range of services, without change to the services themselves, that are not possible to support when a site-oriented view is taken.

A second key concept in Desperanto's design is that it is to be implemented as a "guest" layer, on top of local operating systems, and not as a base or native operating system. We believe this design decision is important when sites participating in the distributed environment are, for the most part, functionally and administratively autonomous, although we have not yet solved the problem of installing Desperanto without change to local operating systems.

Module-Dependent Support

Because of its top-down design, Desperanto is able to support the installation of all types of already existing modules, without change to the module, while site-oriented systems can install only certain kinds of modules without change. Also, Desperanto provides a richer set of primitives for the creation of new distributed modules than is possible with site-oriented designs.

A careful argument is presented in [6] demonstrating that site-dependent designs are not adequate if a system goal is to incorporate already existing modules into the distributed environment without change. Basically, a problem arises while retrofitting modules because some of them generate and save their own objects for subsequent use. In order for such a module to be installed properly, the distributed software has to have knowledge of these module-dependent objects. Therefore, module-dependent information must be available for an invocation of the module by a distributed user. Distributed software designs that are strictly site-dependent cannot accommodate such modules without change.

In addition to offering greater flexibility in installing existing modules, Desperanto also offers more powerful primitives for installing new distributed modules. As an example of these, consider a module that acts as a "filter" in that it accepts a standard input and produces a standard output. In UNIX, the "piping" of filters is supported as an operating system primitive. Basically, a pipe is a temporary unnamed file to which one filter writes and from which another filter reads. Thus, the user can name one input file, a string of filters and only one output file and is relieved of naming temporary files and invoking each filter

separately. UNIX is able to offer the piping facility only because it rigidly enforces the uniform format of the standard input and output of all filters.

Now in a distributed environment where a much larger variety of filters may be available and where enforcing a uniform format for input and output may be difficult, if not impossible, such a piping facility could not be supported by the distributed software unless module-dependent information were available to suitably convert one filter's output to another's input.

Thus, the need for module-dependent information is clearly established. For efficiency reasons it is important to recognize that some modules may not need any distributed support other than that which is site-dependent (see [6] for an example of these) and some modules may fall into certain classes that can be supported by a single module interface. But the capability for supporting individual modules rather than individual sites in a distributed environment cannot be abandoned without loss of flexibility and power.

Reliance on Local Operating System Services

Because of its reliance on traditional local operating system services as a "guest", Desperanto is able to provide a distributed environment to modules on a selective basis. The alternative to this approach is to design a distributed operating system from scratch, managing all resources on each heterogeneous site such as files, CPU's, memory and so on, as a "native" to replace the local operating system. We believe our design offers two significant advantages when participating sites are functionally and administratively autonomous.

The first advantage comes because Desperanto only has access to modules which have been installed in the distributed environment. Modules which have not been installed are never known or used by Desperanto. Since we envision autonomous sites, this flexibility to selectively participate in the distributed environment is highly desirable from a management point of view.

The second advantage is a performance one. Since we view most sites which install modules in our distributed environment to be already operating in a local environment, with well established performance characteristics, it is highly desirable not to degrade the performance of executing local system and application software. A native distributed operating system can never guarantee equal (or better) performance characteristics on all heterogeneous sites on which it is installed. Our approach affects the performance of non-participating local modules only insofar as the Desperanto monitor is an added system module, vying for system resources with all other modules. Thus, all other local operations can perform virtually the same as they did before Desperanto is introduced.

The Desperanto System Model

Desperanto is viewed as consisting of a set of distributed, sharable modules (see Figure 1) which may have already been developed by owner programmers. These modules may be said to have been "programmed-in-the-small" in the sense of DeRemer [1]. A module is a set of data objects (possibly empty) and a non-empty set of associated operations. Further, it may have other components associated with it, like descriptive text. A module may provide operations on objects to other modules and/or require operations from other modules. These operations (e.g., READ, UPDATE, SEARCH) may be thought of as services. A module interface (MI) specifies each service that the module provides and requires. This specification is given by the programmer who is responsible for programming-in-the-large.

Desperanto can be viewed as effecting a virtual procedure call for modules that already exist and as an asynchronous message passing facility for newly created distributed modules. The vast majority of modules that have been coded to run on a single site will provide and require operations by way of procedure calls. If a given local operation deals with remote

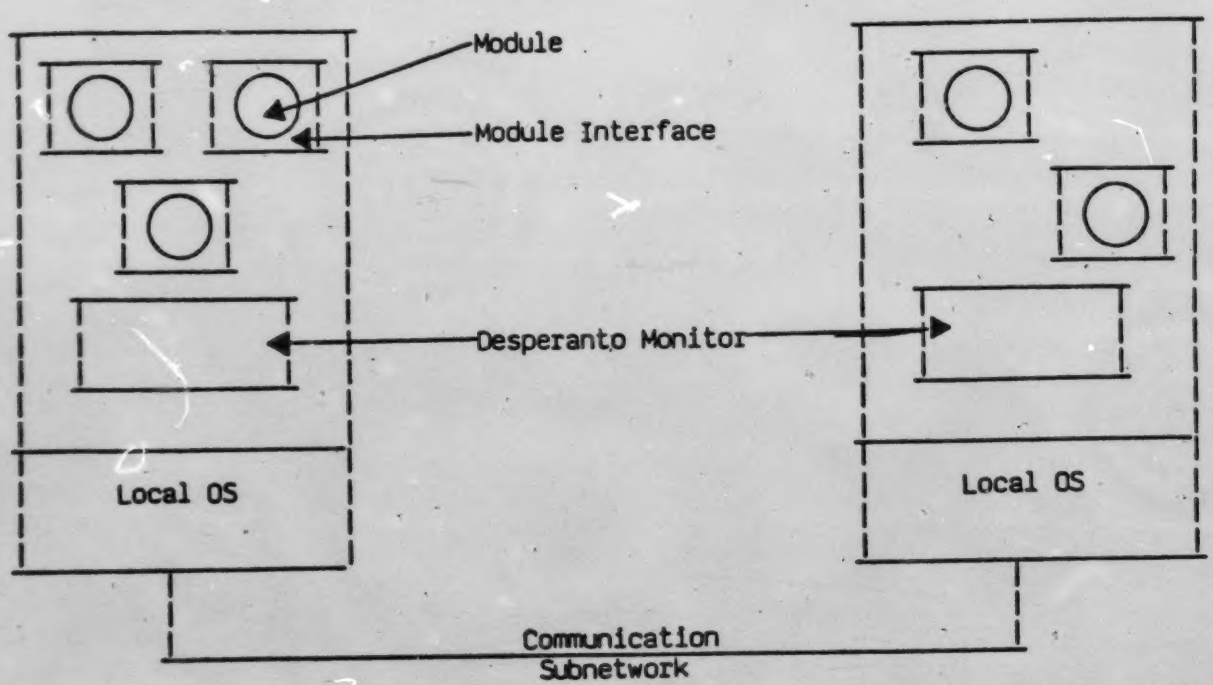

Figure 1. Desperanto Distributed Computer System Model

objects, then Desperanto intervenes to effect the virtual call. New modules may likely be written to take advantage of the distributed environment and to provide and require operations asynchronously. In this case Desperanto supports message passing.

Research and Development Issues

Many research and development issues are being addressed in the design and eventual implementation of a prototype Desperanto. These are briefly outlined here to indicate the scope of the project.

Runtime Components

The runtime components required to support these distributed functions of Desperanto are the monitor, the module interface and the server. Spanning these components are design issues common to all of Desperanto. These include naming, reliability, protection, measurement, testing, and debugging.

Desperanto's Interface to the Outside World

Desperanto's interface components include a virtual local operating system interface, a virtual communications subnetwork interface and a programmer-in-the-large programming and development environment.

The Specification of Desperanto Itself

Because Desperanto is a very large software development project, we felt it would be highly advantageous to use a formalized system development methodology for Desperanto itself. In addition to the fact that the system is fairly complex, the research project is staffed with a constantly changing group of people with diverse backgrounds. So, there is a need for good coordination and effective communication.

Discussion and Conclusion

The Desperanto research project is addressing the design and eventual implementation of software support for distributed processing. Two important design decisions set this work apart from other research efforts. Our module-oriented view gives Desperanto more power and flexibility than site-oriented designs, and our presence as a guest layer on local operating systems is more suitable for autonomous sites than native or base distributed support. The primary long-term goal of this research project is to investigate the feasibility and practicality of installing a distributed support system without change to currently existing resources.

Acknowledgments

Many graduate students have contributed to Desperanto's design in various degrees. We would like to acknowledge the contributions of J. Alegria, W. Ayen, A. El-Magarmid, F. Gherfal, J. Kuo, P. Maurath, Y. Mohammad-Makki, D. Soni, D. Umbaugh, and M. Watkins.

References

[1] F. DeRemer and H. H. Kron, "Programming-in-the Large Versus Programming-in-the-Small," IEEE Transactions on Software Engineering, Vol. SE-3, No. 2, June 1976, pp. 80-86.

[2] J. A. Feldman, "High Level Programming for Distributed Computing", Communications of the ACM, Vol. 22, No. 6, June 1979, pp. 353-368.

[3] H. C. Forsdick et al, "Operating Systems for Computer Networks", Computer, January 1978, pp. 48-57.

[4] S. R. Kimbleton and R. L. Mandell, "A Perspective on Network Operating Systems", AFIPS National Computer Conference Proceedings, Vol. 45, New York, 1976, pp. 551-559.

[5] B. Liskov, "Programming Methodology", Laboratory for Computer Science Progress Report, MIT, July 1979-June 1980, pp. 175-215.

[6] S. A. Mamrak, "Installing Existing Tools in a Distributed Processing Environment", submitted to Workshop on Foundations of Software Technology, Indian Institute of Science, Bangalore, India, December 1981.

[7] S. A. Mamrak and T. J. Berk, "The Desperanto Research Project", OSU-CIS Research Center Technical Report (OSU-CISRC-TR-81-2), February 1981.

[8] D. L. Mills, "An Overview of the Distributed Computer Network", AFIPS National Computer Conference Proceedings, Vol. 45, New York, 1976, pp. 523-531.

[9] R. H. Thomas, "A Resource Sharing Executive for the ARPANET", AFIPS National Computer Conference Proceedings, Vol. 42, 1973, pp. 155-164.

[10] R. W. Watson and J. G. Fletcher, "An Architecture for Support of Network Operating System Services", Computer Networks, Vol. 4, No. 1, February 1980, pp. 33-49.

[11] J. E. White, "A High-Level Framework for Network-Based Resource Sharing", AFIPS National Computer Conference Proceedings, Vol. 45, New York, 1976, pp. 561-570.

2. DEVELOPMENT OF A HIGH LEVEL SPECIFICATION LANGUAGE TO ENFORCE RESOURCE SHARING

Faculty D. W. Leinbaugh

Introduction

An important activity in any multiple user system is resource sharing. This is true whether the users are different peoples' jobs, different processes in the same job, or the processes that make up a system.

Many schemes have been proposed and developed to aid in resource sharing. Monitors [1], object managers [2], and serializers [3] were designed primarily to enforce cooperation among users sharing resources. These schemes provide primitives and language structures which make it relatively easy to write code to enforce the necessary rules and desired policies upon resource sharing. However, these schemes require the writing of programs to provide the necessary synchronization.

This author [4] has designed a prototype high level specification language and system to specify the resource sharing rules needed and policies wanted. He has also described how to automatically construct the code to enforce the specifications. The advantages of this approach are clear. Since the rules and policies are specified directly, it is known exactly what they are and that they are enforced. Ramamritham and Keller [5] concurrently with and independent of this work, attacked the same problem. The specification language they describe is not as concise and consequently is not as easy to read or use. Also, state variables are handled entirely differently.

The specification of resource sharing is given in three independent components. First, resource constraint rules that are used to determine if an additional request can be accepted by the resource and still maintain resource consistency and correct servicing of requests. Second, request ordering policy to determine which of several acceptable requests will be next serviced. Third, modifications to the ordering policy to avoid endless waits by some requests and hence avoid starvation of the processes waiting on the completion of these requests.

The ability to specify resource sharing in three independent components has great advantages. It reduces the problem of resource sharing into simpler components, making it easier to correctly specify each. These are natural divisions in the sharing problem and allow a person to more clearly deal with each separately.

Examples

Figure 1 shows the overall scheduling strategy to enforce high level specification of resource sharing. Some examples will make clear the potential of this approach. These examples illustrate the concise

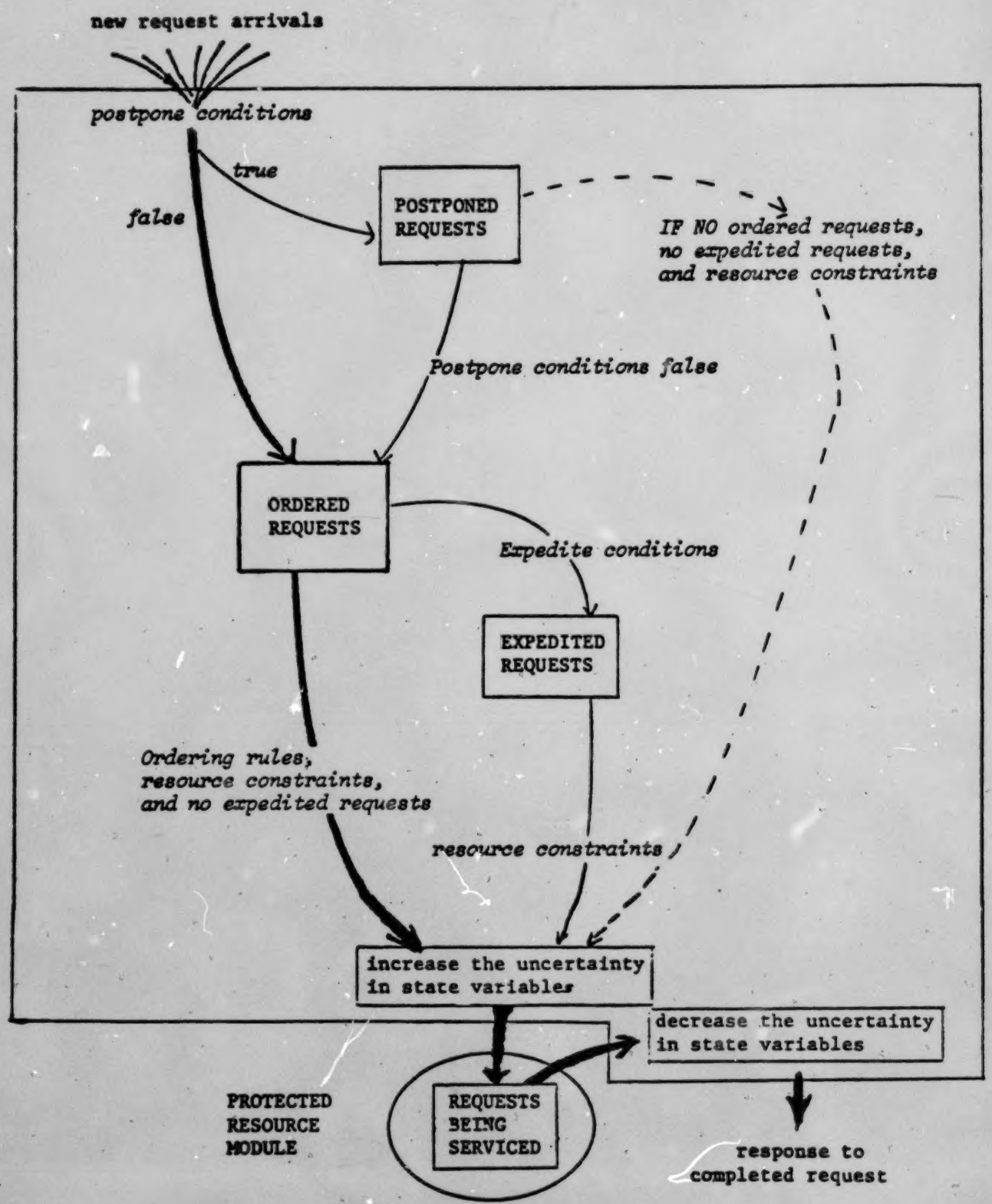

Figure 1. Overall Scheduling Strategy

and straightforward manner in which resource sharing can be specified and enforced.

Figure 2 is a specification of the classical producer/consumer problem. The resource can hold up to 10 items. An insert request message to the insert routine adds another item into the resource and a remove request message removes an item from it.

The constraints upon the resource are specified by RESOURCE CONSTRAINTS. At most one insert request and one remove request can be serviced at the same time. The maximum number of items that can be placed in the resource is 10 and the minimum number is 0. To use the constraints, the preconditions for each type of request are derived. For the case of an insert request, the preconditions are that there are less than 10 items already saved and no other insert request is receiving service. The number of items is kept track of in the scheduling module through the use of the local state variable #items. The PROCESSING clause indicates that during service of an insert request, the number of items is increased by 1 and during service of a remove request, the number of items decreases by 1. There is, however, uncertainty as to exactly when these changes occur. #items is kept as a range of possible values. For example, if there were 9 items and both a remove and insert request were receiving service,

```
DECLARE STATE VARIABLES                  #items          INITIALLY 0

REQUEST DECLARATIONS
            REQUEST FIELDS               type            CHARACTER(1)
                                         item            CHARACTER(39)

            insertitem HAS               type = 'I'
            removeitem HAS               type = 'R'

PROCESSING
    insertitem  PROCESSED BY  insertroutine
                        UPON SERVICE     #items := #items + 1

    removeitem  PROCESSED BY  removeroutine
                        UPON SERVICE     #items := #items - 1

RESOURCE CONSTRAINTS
    insertitem.ACTIVE  ≤ 1       AND       removeitem.ACTIVE  ≤ 1

AND  0 ≤ #items                  AND       #items ≤ 10

ORDERING
     insertitem  BEFORE  removeitem
```

Figure 2: Producer/Consumer Problem with Producer Priority and Capability to Save 10 Produced Items.

#items is kept as the range [8, 10]. If the remove request completes first the range becomes [8, 9] and when the insert request subsequently completes the range becomes [9, 9].

Ordering indicates that insert requests are to have priority over remove requests. The implementation of this problem results in two first-in-first-out queues, one for inserts and one for removes. The oldest insert request is examined to determine if it can be granted service. If there are no insert requests or they cannot be granted service, then the oldest remove request is examined to determine if it can be granted service.

The reader/writer problem of Figure 3 illustrates other features of the specification language.

The resource constraints allow just one writer or any number of readers. The read requests are given priority for efficiency reasons assuming that read requests performed concurrently can be done more efficiently than read requests done one at a time.

```
REQUEST DECLARATIONS
     REQUEST FIELDS  type  CHARACTER(1)
                     directions CHARACTER(63)
     readrequest  HAS  type = 'R'
     writerequest  HAS  type = 'W'
RESOURCE CONSTRAINTS
     writerequest.ACTIVE ≤ 1  AND  readrequest.ACTIVE = 0
     OR
     writerequest.ACTIVE = 0  AND  readrequest.ACTIVE ≤ ∞
ORDERING  readrequest  BEFORE  writerequest
EXPEDITE  writerequest IF  writerequest.EXPEDITED = 0
                       AND writerequest.ACTIVE = 0
                       AND readrequest.WAITING = 0
```

Figure 3. Reader/Writer Problem: Reader Priority

An expedite condition is needed to prevent starvation of write requests if there is a steady stream of read requests. The EXPEDITE condition specifies to choose a writerequest to be next serviced if no write request is in service (ACTIVE) or is in line to receive service (EXPEDITED) and if all read requests that have arrived have already entered service.

The moving head disk scheduler of Figure 4 illustrates more complex request ordering and another feature to avoid extremely poor service to some requests.

The ordering is based upon values in the requests themselves (the disk address) and is according to a predefined ordering technique (elevator algorithm). A secondary ordering criteria is also given. The requests are therefore organized into pairs of sets of requests. Each pair is for a different address with one set of the pair containing read requests for that address and the other set containing the write requests.

This ordering can still result in no service for a request if newly arrived requests to the same address continue to receive service. POSTPONE removes newly arrived requests from consideration for service as long as the address requested is the address currently being serviced.

```
REQUEST DECLARATIONS
    REQUEST FIELDS type CHARACTER(1)
                   address CHARACTER(6)
                   data CHARACTER(505)
        readrequest HAS type = 'R'
        writerequest HAS type = 'W'
RESOURCE CONSTRAINTS
        readrequest.ACTIVE + writerequest.ACTIVE ≤ 1
ORDERING PRIMARY BY ELEVATOR ON address
        SECONDARY writerequest BEFORE readrequest
POSTPONE readrequest IF THISREQUEST.address = ACTIVE.address
        writerequest IF THISREQUEST.address = ACTIVE.address
```

Figure 4. Moving Head Disk Scheduler

Proposed Work

Much work remains to be done to make this a viable tool.

An important capability is to allow for coordination of requests to several resources rather than to a single resource. This includes multiple copies of the identical resource as well as resources of differing types or characteristics. An example is requests for workspace on any of several disks. The resource scheduler should be able to express the usage on each disk and allocate space in a manner that loads the disks equally or according to some other predetermined policy.

Another needed capability is to permit a single request to require service on several resources or equivalently need to perform several operations. This may be several sequential operations that must be performed. Coordination may be required so two operations that can be performed in parallel are completed before a third operation is done. It may even be that one request must result in operations being carried out on different resources simultaneously.

Within this wider context of additional capabilities and even for the prototype system completed, much work needs to be done to make the features truly useful.

It must be determined what kinds of state variables and changes to those variables are needed. The prototype system allows for simple changes for each request serviced. This makes it possible to determine the degree of uncertainty in a state variable and also that the resource might actually be in any of those states. If such changes to the state variables are not adequate to keep track of the resource state, then other mechanisms will have to be analyzed.

It must be determined what kinds of ordering are needed. In the absence of any ordering specification, requests which fit the resource constraints are accepted in a first-come-first-served order. The prototype system allows for priority between resource types. Request ordering can also be specified according to a standard algorithm based upon a value in the request. The standard algorithms built into the prototype system are simple priority and elevator priority. Other standard algorithms may be needed. An additional possibility is to allow the user to specify their own algorithm. This, however, may introduce problems in verifying the correctness of a set of specifications unless the algorithm itself was given as a set of specifications.

Perhaps the most unusual feature of this system and one that has the potential for making it the most useful is the specification of a fairness policy independent of ordering policy and independent of the resource constraints. Of course, they are related in that if the ordering policy or resource constraints are changed, then a different class of requests may be treated unfairly and consequently a different fairness policy is needed to help those requests.

The prototype system provides two separate mechanisms. POSTPONE is used to hold back requests which are responsible for other requests being ignored. EXPEDITE is used to identify requests that are being treated unfairly and schedule them for service next. If there is no better policy for a given problem, the prototype system efficiently provides a count of how many times a request has been bypassed and a request passed over too many times can then be expedited. EXPEDITE, however, makes no allowances to let requests ahead of an expedited request. This could be allowed if in so doing it will not further delay the expedited request.

Implementation efficiency can spell success or failure of the system as a useful tool. The prototype system was designed with efficiency in mind. Sets of requests indistinguishable by the ordering policy are further partitioned into sets indistinguishable by the resource constraints. Therefore, only one request representing an entire set of requests need be examined for each distinguishable set. A similar attention to efficiency must be paid in working the proposed system into a useful tool. Further improvements can be made by only checking those conditions for which some of the terms have changed. The set of specifications supplied may be reducible into simpler conditions.

Bloom [6] has developed general evaluation criteria of modularity, priority, exclusion, expressive power, and ease of use for synchronization systems. The insight needed to design useful features will, however, be gained by compiling an extensive set of synchronization problems and resource constraints.

Many classical synchronization problems can be found in the literature. Other problems can be found or developed from studying operating systems' resource requirements. Combinations of these problems will help in developing features to coordinate requests to several resources and to permit a single request to require more than one resource.

Different features can then be developed and evaluated to express the resource constraints and policies to achieve resource or requestor efficiency and fairness for these synchronization problems.

References

[1] C. A. R. Hoare, "Monitors: An Operating System Structuring Concept", CACM, 17, 549-557, 1974.

[2] R. Bayer, R. M. Graham, and G. Seegmuller (eds.), Operating Systems: An Advanced Course, Lecture Notes in Computer Science, Vol. 60, 262-263, Springer-Verlag, 1978.

[3] C. E. Hewitt and R. R. Atkinson, "Specification and Proof Techniques for Serializers", IEEE Transactions on Software Engineering, SE-5, 1, 10-23, 1979.

[4] D. Leinbaugh, "High Level Specification and Implementation of Resource Sharing", OSU-CIS Research Center Technical Report (OSU-CISRC-TR-81-3), 1981.

[5] K. Ramamritham and R. M. Keller, "Specification and Synthesis of Synchronizers", Proc. 1980 International Conference on Parallel Processing, 311-321, August 1980.

[6] T. Bloom, "Evaluating Synchronization Mechanisms", Proc. of the Seventh Symposium on Operating Systems, 24-32, December 1979.

3. CUSTOMIZABLE WORKSTATION ENVIRONMENT

Faculty J. Ramanathan

Students H. J. Kuo,, C. H. Li, C. J. Shubra, D. Soni

Introduction

We believe that customizable environments are particularly useful in adjusting to the unique and demanding needs of a user community and thereby providing friendly and effective support for the software engineering process. Some examples of customizable tools/environments that have been notably successful are UNIX, SCRIBE, and EMACS. To what extent can experience in developing and maintaining software products be used to customize an environment for profitably supporting the development and maintenance of subsequent software systems? This question fundamentally motivates the TRIAD project described here. The TRIAD environment is 'extremely' customizable thus permitting a project manager to customize the environment and reflect experience culled from other similar projects.

Experience in the development and maintenance of software leads to the design of methodologies for different phases of the software engineering process. Such methodologies attempt to usefully support the programmer's thought process for re-creating only good patterns of programming without limiting creativity. Therefore, to address the above question effectively we propose to design and implement generic tools customizable by a methodology suitable for a given project. Furthermore, the methodologies can be customized to exploit a user group's project oriented experience. For example, a customized methodology may guide the synthesis of a software system using existing 'components'. The significance of our research effort in the design and implementation of tool-box environments customizable by methodologies for the software engineering process is summarized below.

Major issues that must be addressed are:

- A precise notation for describing methodologies must be developed.
- The generic tools must be able to interpret a methodology description and provide support accordingly. That is, the tools must be customizable by the methodology description written using the precise notation.
- A methodology should be customizable by altering its description.
- The environment must interface with the user in a friendly and effective way.

- How should the tools interact with each other to provide an integrated tool box environment?
- The methodologies must be designed and customized considering the human user.
- How should a methodology be customized for a problem domain?

The extent to which the issues have been addressed by the TRIAD project are evident from the characterization of the system given below.

- It has been designed as an integrated collection of tools.
- It 'enforces' methodologies in a friendly manner by requiring the programmer to fill in pertinent 'blank forms' displayed on the screen and later analyzing the blank forms. Help forms explaining how to fill up the blank forms may also be displayed on the screen.
- The generic tools in the tool-box environment are customizable by any of the methodologies in the environment's data base. That is, the programmer selects the methodology most suited or a particular project. The methodology then drives the generic tools which display, analyze, and guide the use of the forms enforcing the methodology. Thus, our environment can provide friendly and application oriented support to the programmer.
- The attributed grammar form model underlying our environment is an extension of the grammar form model which has been used by Ginsburg [2] to model families of grammars. The model provides a precise way for encoding methodologies. The generic tools, which interpret methodology encodings, are analogous to the customizable generic parser. The generic parser can be customized, by the grammar based encoding of a language, in order to enforce the syntax of the language. Similarly, the generic tools can be customized, by the grammar form based encoding of a methodology, in order to provide application oriented support prescribed by the methodology. A methodology encoding may describe languages, but more interestingly, it may describe methodologies for programming-in-the-large (such as DREAM, INTERCOL, SADT, etc.), for programming-in-the-small (such as JACKSON, WELLMADE, etc.), and others oriented towards specific problem domains.
- It provides systematic ways for customizing methodologies. That is, the encoding for a customized methodology can be derived systematically from the encoding of the original methodology. The power of the customization functions depends on the notation used for specifying the customization and the model on which the notation is based.

What is a Methodology and How is it Enforced?

Most methodologies which support the software engineering process are based on the dual notions of refinement and abstraction. Thus the result of applying a methodology can be naturally represented as a tree that we call a refinement tree. In addition, methodologies generally have some of the aspects listed below. We follow each aspect by the user's view of the aspect when our generic tool-box environment is customized by the Jackson methodology. We choose to illustrate the applicability of the model using the Jackson design technique only because it explicitly identifies all the aspects given below.

- Conceptual models which must be used by the programmer to organize the problem at hand in order to achieve the overall program architecture prescribed by the methodology.

  The blank forms shown in Figure 1 have underlying concepts which are a part of Jackson methodology. The first concept requires the programmer to specify the structure of the input and output, and the program driver. By requiring the programmer to fill in the blank forms such as those illustrated in Figure 1, the programmer is encouraged by our environment to organize the problem according to the concepts of the methodology. Forms can be refined using other forms, thus creating a refinement tree.

- Notation prescribed by the methodology which the programmer uses to record the use of concepts both during the development process as well as during maintenance. For convenience, the notation may be designed to reflect the concepts.

  In our environment the notation is used to fill in or interpret a blank form and create an instance of the concept. The Jackson methodology notation uses alternate/sequence/iteration symbols as well as a pseudo code. Note that the alternate/sequence/iteration symbols also reflect the concepts as illustrated in Figure 1.

- Techniques which include management and sequencing functions that the programmer may apply (perhaps using tools) to the documents thus far developed and then determine the next step to take for development or maintenance. These techniques, based on past experience, attempt to reduce the number of possible solutions to the problem at hand.

  In our environment, the techniques suggest the proper forms to be completed by the programmer. A technique suggested by Jack son is to complete the forms which refine the input and output structures before refining the program body.

- Analysis activities based on a global view of the product developed thus far. Analysis is used to answer questions about the methodological aspects or the structure of the product itself. This activity is most likely to be supported by analysis tools of some sort. The results of an analysis may determine what techniques can be applied.

  In our tool-box environment various tools analyze the filled forms to aid the programmer in error and anomaly detection and other verification activities.

A model for encoding methodologies must be able to represent all the aspects discussed above. Attribute grammar forms provide a powerful notation for describing a variety of methodologies for various phases of the software life cycle.

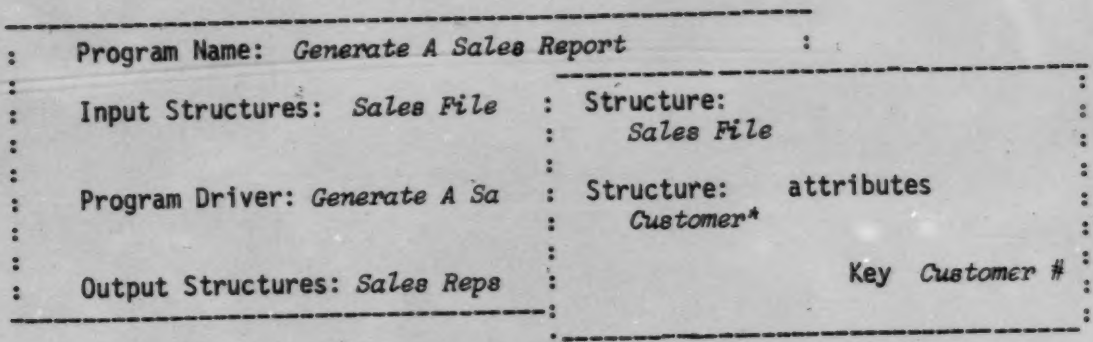

Figure 1. A Hierarchy of two Forms with Interpretations
The blank forms are interpreted (filled) by the user. The interpretations are shown in italics.

Program Developer Tool of the Triad Tool-Box

To illustrate the nature of tools within the tool-box, we briefly describe a specific tool used to develop documents. The tool is called the TRIAD DEVELOPER. A prototype of TRIAD DEV has been implemented in LISP under the DEC TOPS 20 monitor [4]. Other tools, such as the analyzer for analyzing forms, have been designed [1] but not yet implemented. The design of the TRIAD DEV is organized based upon the grammar form model. The productions of the grammar form are displayed to the programmer as blank forms. Thus the programmer does not have to be aware of the details of the formal model underlying the environment. For example, the production underlying the form in Figure 1 is

```
<program name> --> <input structures>
                   <program driver>
                   <output structures>
```

Productions such as this one underlie forms filled by the programmer in order to obtain the following design document.

The design document given in Figure 2 is the result of filling the blank forms for the Transaction Processing Methodology. The symbols in the blank forms guide the programmer's thought process. The interpretations used by the programmer for filling the blank forms are given in italics. TRIAD DEV is unique in that it provides editing functions oriented towards high level concepts. For example the program maintainer can type replace match to change the WHILE loop under match. Since maintenance will generally be required either on the match code or the no match code, editing functions for retrieving and manipulating these text segments are very useful.

There are many instances of the transaction processing problem --- the update of a Master Product File using Product Update Transactions, or the update of Customer records using Customer Payment transactions etcetera. Since the productions underlying the forms enforce a customized version of Jackson methodology as applied to the transaction processing problem domain,

Transaction Program Name: *Product Master File Update*

Input Structures

Name of Transaction File: *Product Update File (PUF)*
Tkey: *Product Number*
Key Ordering: *Ascending*
Name of Master File: *Product File (PF)*
Mkey: *Product Number*
Key Ordering: *Ascending*

Program Driver

Initial Part: *Open PUF, PF; Initialize all variables;*
WHILE ~eof in transaction file:
DO form unit and process it

IF tkey>mkey
THEN flush master
process master records
that are not needed: *Write PF*
read new master: *Read PF*

ELSE process transaction

IF tkey<mkey
THEN no match
WHILE same transaction key
DO no match process: *Print invalid transaction*
read next transaction: *Read PUF*
OD
ELSE match
WHILE same transaction key
DO validation process
validation step1: *Check if transaction file is valid*
validation step2: *Check if product description in PUF record matches PF record*
OD
OD
final part:
Output Structure
END

Explanation: This document is the result of filling the blank forms in the proper order. Programmer's responses to the blank forms are given in italics.

Figure 2. Interpretation of the Transaction Processing Methodology Forms for Product Master File Update.

all transaction processing programs developed using the forms have the same underlying 'good' pattern prescribed by Jackson.

The use of encoded concepts during program development can ensure that there is less variation in the programs developed for a given problem. This is supported by our analysis of programs for a given instance of the transaction processing problem. These programs were turned in by students taking the file management course taught at Ohio State University. The students were already versed in structured programming. By examining fourteen different programs turned in for the master file update problem, we found that the fourteen programs had a significant variety of control structures [3]! This is despite the fact that all the students attempted to use Jackson methodology. The use of the transaction processing methodology, supported by a semi-automatic program development system ensures that only one control structure is actually generated and thus there is less variation in the transaction processing programs developed. In other words the consistent application of a methodology results in a more maintainable final product.

Concluding Remarks

Methodologies for the software engineering process constantly evolve to suit the needs of a user community. Furthermore, methodologies are generally imprecisely defined. Thus there is a great deal of variation in programs resulting from the manual application of a methodology and programs often remain unreliable and difficult to maintain. This suggests the need for well-tailored methodologies which guide the programmer's thought process for recreating only 'good', standard patterns or techniques of programming. In other words environments should support well-tailored methodologies which encode our collective programming experience, and the degree of effectiveness of these methodologies in improving software quality should be demonstrated.

The TRIAD tool-box environment will be customizable by methodologies and will be instrumented to gather data. In addition to addressing research issues in the design and implementation of customizable environments, the TRIAD project will gather data to study the use of methodologies by programmers and the impact of different methodologies on software quality.

References

[1] J. Arthur and J. Ramanathan, "Selective Program Analyzers", IEEE Transactions on Software Engineering, January 1981.

[2] S. Ginsburg, "A Survey of Grammar Forms - 1977", Sixth International Symposium on Math. Foundations of Computer Science, TatranskaLomnica, Czechoslovakia, 5-9, 1977.

[3] J. Ramanathan and C. J. Shubra, "The Modeling of Problem Domains for Driving Program Development Systems", Eight Annual Symposium on the Principles of Programming Languages, January 1981.

[4] J. Ramanathan and H. J. Kuo, "Concept Based Tool for Standardized Program Development", COMPSAC, 1981.

4. CURRENT RESEARCH IN COMPUTER GRAPHICS

Faculty F. Crow

Students M. Howard
Win Bo

Abstract

Sponsored research in computer graphics currently encompasses projects to (1) provide more flexible algorithms for image generation, and 2) improve techniques for removing the side effects of digital image generation (known as "aliasing").

(1) To generate images more flexibly, a supervisory process is used to distribute picture-generation tasks to heterogeneous subprocesses. Significant advantages accrue by tailoring the subprocesses to their tasks. In particular, scan conversion algorithms tailored to different surface types may be used in the same image.

(2) Sufficient reduction of aliasing effects under all possible conditions remains difficult except by expensive (brute-force, high-resolution) techniques. Some experiences in trying to discover what level of effort is necessary to achieve "acceptable" images have led us to the question of when overlapping and abutting surfaces need to be distinguished within a pixel. Four paradigms for rendering order are addressed in this context.

A More Flexible Image Generation Environment

Introduction -

Current software display systems are either too restrictive, e.g. allowing display only of surfaces defined as polygons, or demand too much of the operator, e.g. determining priority. The most heavily used display algorithms have been monolithic software systems based on a single type of object description. In order to make very complicated images with such systems, enormous computing resources have been necessary to sort all surface elements to scan-order or priority order. Where such systems have been expanded to include additional surface types, the cost of integrating the new code has been inconvenient at best.

An alternative approach based on small, single-purpose display programs has been used at some sites. This approach cuts the cost of integrating new display techniques to a minimum. However, in order to make a complex image a complex command sequence is necessary, forcing the user of the display programs to have most of the skills of an experienced programmer. Furthermore, the user must be unnecessarily intimately involved in the task of image generation, a task which proceeds automatically in the monolithic systems.

In the interests of providing an environment with maximum flexibility for production of the widest possible range of imagery and easy introduction of experimental display algorithms, I am currently exploring a middle ground in which a supervisory process serves the integrating function of the user in the second scenario. This involves implementing a system which tries to separate the process of image generation into two distinct phases: scene analysis and object rendering.

During scene analysis interobject interactions are determined solely on the basis of crude estimations of the size and position of objects making up the scene. Information from the scene analysis is then used to determine how to generate the image. Object priority, used to define the order in which objects are rendered may be found. Object interaction such as intersection, cast shadows, and proximity for color interaction may all be determined by the scene analysis.

Object rendering may be done by a number of independent processes in the case of non-interacting objects. The independent processes may be prioritized by execution sequence on a single-processor system or by interprocess signals on a multiprocessor system. Object interactions may be resolved by several methods including combining each interacting set of objects into a single rendering process.

A Multitasking Execution Environment

To even consider building the kind of software described here a flexible operating system which allows a user process to spawn and control daughter processes is imperative. While more and more such systems are available, commercial operating systems frequently do not anticipate such use. We are using the UNIX operating system with VAX extensions from Berkeley to support this work. While the existing software provides a reasonable environment, planned extensions promise to make this system even better suited to the task.

The primary task of the supervisory process is to determine which objects overlap in the image and then determine the depth order within a group of overlapping objects. Any of a number of extant priority algorithms would suffice for this purpose. However, the elements to be compared here are objects (or clusters of related objects). The intention is that there should never be so many objects compared in one process that the order of complexity of the comparison process or the amount of available memory forms an important restriction.

The supervisory process reads a description of a scene, giving posi-

tions and orientations of objects, light source specifications, and view parameters. Succeeding scenes may then be described incrementally, providing only the changes from the previous scene. This form of description, while bulkier than some form of key-frame description, provides a well-defined description of frame-to-frame changes while freeing the supervisory program from the tasks of interpolation between key frames. Interpolation is another aspect of animation subject to experimentation and is thus better done elsewhere. The supervisory image generation process is being kept as simple as possible.

The incremental frame-to-frame description allows the use of whatever coherence exists between frames to be put to good use. For example, if the view parameters are unchanged then only moving objects change the image. In such cases, only those parts of the picture overlapping the changing objects need to be updated. The supervisory process may reset clipping parameters to limit the computation to dynamic areas of the image.

The scene description allows some information on object interactions to be specified. For example, calculations can be much improved if it is known that one object is supported by another or, more generally, that two objects are in contact but not interpenetrating. Furthermore, intentionally intersecting objects may be specified. This allows the process to precalculate static intersections and to warn the user of perhaps unintended ones. Shape changes may also be specified as an interpolation factor to be applied to two topologically equivalent objects.

Slave Processes

Independent slave processes can do the actual work of image generation including all operations on individual surface elements (i.e., clipping, shading, scan conversion). As long as individual objects do not interact, the slave processes can be specialized for efficiency in rendering one particular kind of surface. The most obvious example would be the sphere.

There have been a few published algorithms for rendering images consisting solely of spheres. These algorithms were significantly superior to polygon algorithms for the purpose and have proven useful in several areas. However, sphere-specialized algorithms have been of limited general-purpose use since other types of surfaces cannot be included in the images they produce.

In general, an image may be made somewhat more efficiently by algorithms tailored to individual surfaces. The big advantage, however, lies in the much greater simplicity of tailored algorithms. This simplicity allows us to consider implementing the algorithms in other ways (e.g. microcode or hardware) which would be impractical for more complex algorithms.

For example, an efficient display algorithm for polygons usually has to make priority decisions based either on local comparisons or a very complicated sort scheme with order n-squared complexity. Hardware shaders or tilers can speed up execution of such algorithms by factors of two or three. But the complexity of the algorithms prevents serious consideration of further hardware translation except in the largest of undertakings.

The bulk of polygonally-described objects may be sorted by much simpler means. A simple ordering by average depth, for example, is frequently sufficient. A slave process tailored to such objects could do the bulk of the work in most polygon images. Virtually all computation in such an algorithm involves transformations (matrix multiplies), clipping, shading (dot products) and scan conversion (tiler). The sort may be handled by a sufficiently high-resolution bucket sort. In short, no polygon need be considered more than once by any stage of the algorithms, satisfying the conditions for implementation by a pipeline of very simple processes.

Preliminary investigations indicate that removing the need to worry about intersections can similarly simplify algorithms for other types of surfaces. The question then becomes what to do when an object intersects itself or two or more objects cannot be separated by the supervisory process. Obviously, we can retrench and use a last-resort algorithm which handles such surfaces at a loss in efficiency. However, if we can recover detailed intersection information from the last-resort process, then subsequent images may be made more efficiently.

For example, tests by the supervisory process may show that two nearby objects intersect. The last-resort process will find those intersections if they exist or indicate that there are no such intersections. In either case if the objects don't move relative to one another, the information can remove the need to use the last-resort process in succeeding images. The objects may be declared non-intersecting or may be modified to resolve the intersections.

Some question exists as to the form of the last-resort algorithm. Since surfaces of many different formulations may coexist, there can be great difficulty in trying to resolve intersections. Initially, the last-resort algorithm is an existing polygon algorithm. All data types used will have an associated algorithm for expansion of the surface description to polygons for use with the last-resort process.

Experience in Anti-Aliasing

Introduction

The term "aliasing" has been used for some time to refer to the deleterious effects of improperly representing an image as a regular array of discrete dots. These effects are seen as jaggedness along edges, inconsistent size and shape in small objects, and randomly disappearing details. The term "aliasing" derives from the effect caused by looking at a signal at an inadequate number of regular intervals. A 12 cycle signal looked at ("sampled") at 10 regular intervals appears identical to a 2 cycle signal over the same space and sampled at the same intervals. Thus the 2 cycle signal is an "alias" of the 12 cycle signal.

An inadequate sampling interval when synthesizing digital images causes misrepresentation of the local positions of the edges which characterize the image. Unfortunately, technological limits and standardized equipment prevent most of us from choosing our sampling interval. Therefore we are left trying to make the best of what we have. If a display is viewed from an adequate distance, details the size of a pixel cannot be resolved. Therefore,

variation in intensity over a few pixels can be used to "suggest" the position of an edge. By such means it is technically feasible to make an image which is indistinguishable from one of much greater resolution.

The trick is to achieve the same level of intensity at each pixel as would be seen over the equivalent small area in a higher-resolution image. This can be done by properly considering all the details which contribute to each such small area.

In theory, aliasing cannot be entirely eliminated in a finite image. However, it can be diminished to the point where it is undetectable. The means for diminishing aliasing involve increasing the amount of information upon which the image is based. The mosaic of dots making up an image represent samples from a conceptual scene defined by the input data. The information content of the image can be improved by increasing the number of samples or by taking more information into account when taking each sample.

The issues to be decided, then, are how many samples are adequate or how much information should be considered in taking each sample. Unfortunately, different features in an image require differing amounts of care to hide aliasing. Broad areas in which the shade changes slowly obviously can be reproduced with very little information. On the other hand, areas with sharp detail carry a much greater density of information and must be treated more carefully. Of course, areas with extremely fine detail may not be reproducible, given the display resolution. In such cases, the proper representation is a smudge of the correct intensity.

Currently, practical images have large regions of sparse detail. It is therefore inefficient to use brute force techniques, computing the entire image at a much higher resolution then averaging many samples into one displayed pixel. However, this may not be true for future images of much greater complexity. For the moment, algorithms which treat detailed areas with special care remain important enough for study.

Generally, the pixels which need special care are those which contain part of a surface edge. The color of such pixels must be calculated as an average of the local colors of the surfaces on either side of the edge. If the pixel is considered to represent a very small image from the scene being generated, then the color of each surface visible within that image must contribute to the color of the pixel. The pixel color is determined by summing all the contributing surface colors, each weighted by the area it subtends within the pixel. Thus, where an edge passes through a pixel, the surface colors on either side of the edge are blended by a weighted average.

The advent of the inexpensive frame buffer has made viable a number of hidden-surface algorithms which calculate the image in an order dictated by the features of the image rather than the structure of the display (as scan-order algorithms do). A characteristic of such algorithms is that each surface element (polygon, patch, etc.) is scan-converted independently. Pixel-by-pixel comparisons between surface elements are limited to comparisons of intensity, (sometimes) depth, and whatever other information one can afford to store for every pixel. The following sections will show how surface color blending can be done in the context of frame buffer algorithms.

Rendering Order

Nearly all scene generation algorithms fall into one of four categories based on the order in which surfaces are written to the display. (1) Surfaces written in depth-order rearmost first (the "painter's algorithm") (2) Surfaces written in depth-order frontmost first. (3) Surfaces written in scan order. (4) Surfaces written in any order (depth buffer). Exceptions to this classification are the Warnock algorithm and an algorithm invented by I. E. Sutherland. Both these algorithms write in a lateral order (across the screen) determined by scene features. However, for the present purpose, they may be classified with the depth-ordered frontmost-first algorithms.

Algorithms in the first two categories first order surface elements by priority then write each surface element to the display independently. Complete information is available only for the surface element being rendered. Where two surfaces partially cover the same pixel, there is no way to determine whether the surfaces overlap or not. Obviously, the proper intensity is often different for the two cases. The scan-order algorithms, on the other hand, determine surface priority for each scan segment in turn. Thus, all surfaces affecting a given pixel must be known to the algorithm at the time the pixel intensity is computed. In this case, it is possible to resolve distinctions between overlapping and non-overlapping surfaces. The depth-buffer algorithms offer the same problems as the priority algorithms and offer some additional difficulties of their own.

Frequently, it makes no visual difference whether the intensity has been computed absolutely correctly where surface edges come together in a pixel. However, the intensity must be computed in a consistent manner to avoid jagged edges. In the following, methods for anti-aliasing in the face of inadequate information are presented, with results where they are available.

Rear-to-Front Order

Where surfaces are rendered in depth order, rearmost first, the edges of surfaces must be blended with previously written pixel intensities. Where surface elements are meant to blend together to form a continuous surface, arbitrary blending is insufficient. The first surface of an abutting pair will be blended with the background. The other surface will in turn be blended with the first blend. The effect is to reveal the "seams" of the surface. What is desired is a blend of the colors of the two surfaces involved.

In the case where the surface elements are to join smoothly, aliasing poses no problems. Such edges may be ignored. The problem with this lies in determining which edges need special anti-aliasing treatment and which don't. For smoothly shaded polygonal surfaces the solution is relatively simple. The edges joining two front-facing polygons need no special treatment. The edges joining a back-facing polygon with a front-facing one and the edges belonging to only one polygon need anti-aliasing. Algorithms for rendering higher order surfaces often incorporate some technique for determining silhouette points, where the surface curves back on itself. Anti-aliasing can be limited to the silhouette.

Where adjoining surface elements are to join at a crease in the surface,

anti-aliasing is necessary if there is significant contrast across the edge. In these cases, it is sometimes possible just to let the background show through the seam. This will just enhance the edge. However, if there are high-contrast features in the background behind the crease, those features will be seen, especially in animation. Therefore, yet another technique is called for.

For joining surfaces at a crease, let the first surface written entirely cover pixels along the effected edge. The algorithm must ensure that adjacent surfaces will overlap by one pixel everywhere. The second surface may then be blended along the abutting edge. Since the blended pixels will have been set to the color of the first surface, the proper blend of just the colors of the two abutting surfaces will result.

This latter technique requires an expanded description for surfaces which are to be treated this way. It must be possible to conveniently find all the neighbors of a given surface element. With this additional information, each neighboring surface edge may be tagged when a given surface element is written to the display. When an edge is written, it is blended where the adjoining surface element has been written and not blended where the adjoining surface has not yet been written.

It remains to determine what to do with edges which affect the same pixel but which do not belong to the same surface. In these cases, I consider it impractical to try to recover the data describing the earlier edge in order to properly blend it with the latter. In practice, edges of separated surfaces rarely adjoin in such a way as to cause a problem. Each surface edge may be blended with whatever came before it. Problems will occur where two surfaces are intended to abut, hiding any features behind the abutting edge. Lesser problems will occur where overlapping surface edges affect the same pixel. The color of such pixels will be distorted in the direction of the hidden surface. However, this is an effect which should bother virtually nobody.

Various suggestions have been made for encoding information in extra bits of the frame buffer memory to indicate what part of the area included by a pixel has been covered. I don't believe that the extra work involved would provide any significant benefit. However, such methods can be useful for a different rendering order.

Front-to-Rear

There are advantages to rendering closer surfaces first. For example, when the cost of shading each pixel is large, the computations can be avoided where it can be determined that the pixel is already covered. It is also possible to avoid the problems caused by abutting surfaces in the previous section without referencing more than one surface at a time.

Front to rear algorithms must use some means for determining what pixels have been covered by previous surfaces. One approach is to divide the screen along surface edges and clip succeeding surface elements by previously rendered edges. This approach has disadvantages, however. It works fine for polygonal surfaces but does not extend to direct rendering of high-

er-order surfaces. It has an unacceptable order n-squared complexity since each surface element must be compared with a large portion of all previously rendered edges. It requires storing a description of all rendered edges which must be frequently referenced (bad for very complex pictures). An alternative method eliminates these disadvantages at the cost of using some extra frame buffer bits.

The alternative is to use frame buffer bits for what I call an "occlusion buffer". The occlusion buffer stores the percentage of each pixel area which has already been covered by a surface. This concept was used by Catmull although he didn't use the same term to describe it.

Using an occlusion buffer, surfaces are rendered front to rear. As each surface is rendered the occlusion buffer is loaded with numbers indicating how much pixel area is covered. In a simple picture, most entries will indicate complete coverage. Edges, however, will only partially cover. When a second surface is written to a partially-covered pixel, it is assumed that the surfaces abut. Therefore, the existing coverage is used to weight an average of the colors of the two surfaces at that pixel and is then summed with the coverage of the new surface and restored to the occlusion buffer.

When an occlusion buffer value overflows, it must be set to indicate full coverage. Fully covered pixels are then ignored in further computations. As the last step, the background color (or background scene) must be blended with the computed scene using the occlusion buffer to indicate how to use the background.

This scheme works very well in most situations where reasonably isolated objects are depicted. In such situations, surface elements sharing the same pixel are nearly always abutting. Only where different objects or concavities exist are overlapping surfaces within a pixel likely. Furthermore, only where these overlapping surfaces are aligned so that they appear to be a single edge is there a noticeable effect. That tends to be unusual. However, there are situations where such arrangements appear.

Depth Buffer

In the case of the depth buffer it is necessary to handle the case of surfaces being added either in front of or behind existing surfaces. The major drawback to the depth buffer is that there really is no way to handle the case of a surface which lies intermediate, in depth, between two previously written surfaces. The surface must be treated as though it lies behind all surfaces written so far or in front of all such surfaces. This problem has forced escapes into more complicated algorithms when rendering transparent surfaces.

By using the occlusion buffer, most depth-buffer images can be made without severe defects. However, where a surface must be added between two earlier surfaces, the occlusion buffer fails. If a close surface is written first, then a more distant surface behind it, the occlusion buffer will show as completely covered any pixels along edges of the close surface which lie over portions of the distant surface. Thus, it will be impossible to properly blend an intermediate surface behind such edges.

Since it is rare to have three long edges share the same pixel in any reasonable image, perhaps the occlusion buffer can be extended to handle the case described in the preceding paragraph. If one bit of the occlusion buffer is used to indicate complete coverage, the remaining bits could be used to indicate the coverage of the closest surface. This would preserve the necessary information needed to insert a surface behind the frontmost one. The hitch in this lies in the fact that the whole philosophy of the depth buffer algorithm must be broken to make this work. There is no information associated with the pixels in question which can signal the process that the new surface fits between the two surfaces previously stored.

If the depth of the more distant surface is stored with the pixel then the algorithm must overwrite. If the depth of the closer surface is stored then there is no way to know whether the new surface lies in front of the surface lying behind the visible edge. Therefore, to make an expanded occlusion buffer scheme work, some more global information is necessary. Global information could be used in the following way. Wherever a surface changes state from visible to hidden during scanout, blending must occur. If no blending occured in any such transition then information on the depth of neighboring pixels can be used to resolve the ambiguity of the above situation. It is assumed that true surface intersections are not allowed.

Unfortunately, any such scheme violates the most attractive feature of the depth buffer algorithm, its simplicity. The above scheme would only be worthwhile if it could be implemented to cause minimal additional computation except in those cases where it is needed. A check for visibility change would have to be made at every pixel. That could be too much when you consider that very minimal surface sorting steps could be used to remove such cases at the outset.

Scan-Order

Initially, scan-order algorithms were necessary since nobody had frame buffers. However, only those who wish to make images of extravagantly high resolution really need use scan-order algorithms today. On the other hand, in many environments the scan-order algorithms offer a performance advantage since they require no pixel-level comparisons to determine visibility. Where scan segments are, on the average, more than a few pixels in length, the scan-order algorithms can make much better use of local coherence than any of the other approaches. The drawbacks of these algorithms are the horrendous complexity of the code required to make them work and the heavy demands for address space when making complicated images.

The big advantage of scan-order algorithms here is that all necessary information is available to determine the interrelationships of an arbitrary number of edges sharing the same pixel. This information can be used to do a mini-hidden-surface algorithm at the pixel, to calculate the pixel at a higher resolution, or to determine whether one of the first two techniques is really necessary. For simple images, the complicated situations which require the blending of more than two surfaces occur only rarely. In more complicated images involving a lot of small details in which several surfaces share a pixel, I'm not convinced that scan-order algorithms are really practical.

Sampling Window

In calculating areas within a pixel, it has been debated whether it is sufficient to model the image as a mosaic of abutting square pixels. Signal processing wisdom and the experience of television engineers tell us to model the pixels with overlapping areas. The areas should overlap by one-half so that an infinitesimal point in the scene being depicted would appear in four pixel areas. The extent of overlap is based on the size of the central lobe of the ideal low-pass filter for the sampling interval represented by the image structure.

There is no question that the overlapping pixels produce better results in the most difficult cases than do the abutting pixels. However, in less demanding situations the differences are negligible at best. To make comparisions, the same scene was computed at very high resolution using a scan-order algorithm, then averaged to a lower resolution using both pixel models. For the one image, 64 samples were averaged to get the pixel colors, approximating the effect, of convolution with a Fourier window. In the other image, 225 samples were averaged. The samples were weighted more heavily in the middle, approximating the effect of convolution with a Bartlett window.

There was no discernable difference between the two images except in the highlights on the closely spaced columns that appeared in the two pictures. The high contrast of the highlights with the columns and the repetitive nature of that part of the image make a difficult case for anti-aliasing. There is a "braided rope" effect when the abutting pixels are used which is considerably attenuated when overlapping pixels are used.

It remains for the individual application to determine whether the abutting pixel model is sufficient. My experience has been that the overlapping model is necessary only where long, thin, and very bright features are included. Dull images don't need much care.

These highlights in the example represent yet another difficulty in anti-aliasing techniques: they are image details not associated with a surface edge. All selective anti-aliasing schemes are based on providing special attention to those areas where edges pass through a pixel. Another heuristic is needed for determining where there is a small highlight which needs special care.

Conclusions

I have tried to explain some of the conclusions I have reached in experimenting with anti-aliasing schemes over the past several years. My work in this area is still continuing and future developments may leave me with much different conclusions in coming years. It seems likely that for the next decade or so, there will be a great emphasis on algorithms which can be stated simply enough to be conveniently compiled into microcode or hardware. Future work along these lines will be addressing the issue of simplicity more heavily.

ABSTRACTS OF PH.D. DISSERTATIONS 1980-81

CHOU, CHUEN-PU. System Design of the Distributed Loop Database System (DLDBS) The Ohio State University, Spring Quarter 1981.

The Distributed Double-Loop Computer Network (DDLCN) is designed as a fault-tolerant distributed processing system that interconnects midi, mini, and micro computers using a double-loop structure. It serves as a means of investigating fundamental problems in distributed processing and local networking. One of the major distributed services provided by DDLCN is a distributed database system.

In the distributed database system, the user has logically integrated access to a collection of data which is managed on a network of geographically dispersed computers. Users' requests to the database can be satisfied independently of the physical location of the database, database definitions, and the database management system (DBMS). In this dissertation, we present a way to design such a distributed database system for distributed processing systems in general and for DDLCN in particular. We call such a distributed database system the Distributed Loop Database System (DLDBS).

To design a distributed database system is a complex task. A designer has to face many technical problems and design decisions. A centralized single-node database system typically consists of three components: the DBMS, the database directory/dictionary, and the database. In the distributed database system, however, we must be concerned with the design of the distributed DBMC, the management of the database directory, the database distribution, and the distributed database architecture by taking such database distribution into consideration.

The design of the distributed DBMS involves designing three distributed processing algorithms (DPAs): distributed concurrency control, distributed query processing, and reliability. In this dissertation a system organization of the distributed DBMS is first described. Its software components and the inter-relationship among these components are then identified. Two new concurrency control mechanisms, one for fully and one for partially duplicated DLDBS, are presented next. The mechanisms use distributed control and are deadlock free, simple to implement, and robust with respect to failures of communication links and hosts. They do not use global locking, do not reject transactions, and exploit potential concurrency among transactions. Arguments for the correctness of the algorithms are also given. Then a reliability mechanism is presented to ensure the continuing and correct operation of the DLDBS under abnormal conditions. Finally, data definition and manipulation languages for DLDBS are described, and an approach to distributed query processing is also suggested.

In the design of DLDBS we take into consideration special characteristics of DDLCN, which is a local network with broadcast channels. DDLCN has a high bandwidth and low error rate, and most importantly, it supports multi-destination protocols to facilitate efficient implementation of dis-

tributed processing algorithms. The design of DLDBS is intended to be so general that new concepts developed for it can be applied to other distributed database systems. In this dissertation we also demonstrate that it is feasible to integrate database management, computer networking, and distributed processing technologies into a unified system.

LI, CHUNG-MING. Communicating Distributed Processes: A Programming Language Concept for Distributed Systems, The Ohio State University, Winter Quarter 1981.

This dissertation is concerned with the development of a distributed programming language/system, to be called DISLANG, for use in distributed computing systems and computer networks. The purposes of developing DISLANG are 1) to provide the system programmer with a tool for use in implementing distributed system software, and 2) to provide the user with a high-level language for use in writing distributed application programs. We assume a more "realistic" model, meaning that the distributed systems under consideration do not have perfect communication or processing subsystems. We seek a language that can abstract very nicely such inherent characteristics of distributed systems as variable message delays, communication link failures, processing subsystem crashes, data distribution, etc. A new language concept, called Communicating Distributed Processes (CDP), has been proposed to provide language constructs for handling the aforementioned characteristics of distributed systems.

CDP introduces a new language concept called communication/distribution abstraction. All necessary information for process communication, data distribution, and execution parallelism is collected together as a separate component of CDP, called Communicator. Features of process communication, data distribution, and execution parallelism are specified abstractly by using this information, so as to achieve a high level of modularity and a high degree of abstraction.

Operation types are used in CDP to specify the properties of operations at remote nodes in an abstract way so as to achieve communication/distribution abstraction. Operations at remote nodes are handled in the following way: 1) the programmer describes different types of usages for the operations by specifying operation types and 2) the system dynamically handles remote operations by processing the operation types properly.

The design of Communicating Distributed Processes provides a completely new language concept which encapsulates the characteristics of distributed systems, such as distribution of resources, distribution of control, communication delay, and communication failure. In addition, it is consistent with the current trends in programming language design, such as modularity, abstraction, etc.

A high degree of modularity is achieved by using the Communicating Distributed Processes and Communicators, and a high level of abstraction is provided by using operation types. Moreover, the concept of communication/distribution abstraction used in the Communicating Distributed Proc-

esses not only provides a great deal of expressive power for distributed programming but also has the advantages of simpler programming, easier understanding, more flexible modifiability, etc.

MITTAL, SANJAY. Design of a Distributed Medical Diagnosis and Data Base System. The Ohio State University, Summer Quarter 1980.

A methodology is developed for organizing potentially large and diverse bodies of knowledge in a computer system. The methodology is illustrated by the design and implementation of a knowledge-based consultation system called MDX. In the MDX system, different kinds of medical knowledge - diagnostic, anatomical, physiological and clinical - needed for diagnosing diseases in the Cholestasis syndrome are organized in a distributed framework. MDX is organized into three subsystems: A diagnostic system; a patient data base assistant, called PATREC; and a radiology consultant, called RADEX.

The diagnostic knowledge is organized in a Conceptual Hierarchy. Each node in the hierarchy corresponds to a diagnostic state. Associated with each node is a specialist, which contains knowledge for establishing and refining the node. The problem solving strategy of Establish and Refine is implemented as a collection of specialists, communicating via well-defined principles. Some criteria are also developed for determining the distribution of different kinds of knowledge among these specialists. The performance of the system is analyzed by discussing some medical cases in detail.

An important aspect of the methodology described in this work is the emphasis on the organization of auxiliary knowledge, which is not directly involved in the problem solving task, into separate consultants. In particular, the knowledge about medical data entities required for answering questions about the patient data and the anatomical or physiological knowledge required for interpreting radiological information are organized into separate consultants - PATREC and RADEX. The communication between the diagnostic system and the auxiliary consultants is via a query language. Each of these consultants has a conceptual model of the relevant data entities - labtests, signs, symptoms, organs, deformities, etc. - which enables them to make default assumptions and infer information not explicitly stored in the data base. Some criteria and algorithms are developed for making such inferences and assumptions, and for combining information from multiple sources into a composite model. The actual patient data is organized into temporal episodes - clustered around key episodes. Some issues in organizing temporal patient data are also explored.

TSAY, DUEN-PING. MIKE: A Network Operating System for the Distributed Double-Loop Computer Network, The Ohio State University, Spring Quarter 1981.

The proliferation of cost-effective small computers has spurred interests in areas such as distributed processing. This discipline downgrades the importance of processor utilization and emphasizes other goals such as system resource sharing, reliability, and extensibility.

This dissertation proposes the framework and model of a network operating system (NOS) called MIKE and its supporting architecture for use in distributed systems in general and for use in the Distributed Double-Loop Computer Network (DDLCN) in particular. MIKE, which stands for Multicomputer Integrator KErnel, provides system-transparent operation for users and maintains cooperative autonomy among local hosts. Its underlying architecture provides additional hardware/firmware mechanisms to support the logical structure of MIKE.

MIKE incorporates modern operating system design principles to cope with its complexity and vulnerability. These principles include data abstraction, capability-based addressing, and domain-based protection. The use of these concepts can contribute to several characteristics whose presence is essential in a distributed environment. Among these characteristics are extensibility/configurability, reliability/robustness, system transparency, and local autonomy.

An integrated approach is taken to design the NOS model and protocol structure. MIKE is based on the object model and a novel "task" concept, using message passing as an underlying semantic structure. A layered protocol is provided for the distributed system kernel to support the NOS services. This approach provides a versatile network architecture in which system-transparent resource sharing and distributed computing can evolve in a modular fashion.

The architecture of the Loop Interface Unit (LIU) in which MIKE is housed is designed with the explicit purpose of facilitating the implementation and maintenance of a robust NOS for the DDLCN. It is designed in such a way as to narrow the gap between the abstractions called for by the NOS model and the features directly realized by conventional hardware.

The organization of LIU is configured according to the data and control message flow of the MIKE hierarchical framework. Furthermore, to reduce the heavy overhead associated with imposing the logical structure of MIKE onto conventional hardware, LIU is based on a software-directed architecture. Adequate hardware and firmware mechanisms are provided so that modern operating system design principles can be supported at the architectural interface. This integrated hardware and software design approach is required if MIKE is to be implemented efficiently.

In summary, the network operating system adopts modern operating system concepts into its design and has extensive architectural support. The framework of MIKE includes the NOS model and supporting protocol structure.

The Loop Interface Unit (LIU), where MIKE is running, comprises a number of processing units which are configured to optimize the internal message traffic flow. Hardware/firmware mechanisms are augmented to provide a software-directed architecture for MIKE in LIU.

WANG, PONG-SHENG. Computer Architecture for Parallel Execution of High-Level Language Programs, The Ohio State University, Summer Quarter 1980.

A new approach, called Parallel Execution String (PES), is proposed to recognize parallelism in ordinary programs and to represent them in multiprocessor systems for parallel processing. The PES scheme decomposes expressions in such a way that it can eliminate the unnecessary wait before an operation can be started, has minimal intermediate store and fetch of partial results, can minimize the intervention of central control to individual processors, and uses no stacks. The PES approach is then used to recognize the parallelism among a block of statements. A machine organization for executing the programs compiled with the PES approach is proposed. Code generation and optimization techniques are also presented.

The concept of try-ahead processing of IF, REPEAT, WHILE, and LOOP statements is proposed. The representation of these statements for try-ahead processing is presented. With this approach, the delay caused by the evaluation of the boolean expressions in these statements can be greatly reduced.

The architecture of a parallel execution high-level language computer is proposed. In the proposed architecture, the PES approach to parallel processing and the try-ahead processing approach are used. The pipeline effect, parallel processing, and try-ahead processing will result in a system whose performance is much better than that of its individual processors.

The design of a multi-microprocessor system using Am2900 based bit-slice microprocessors is presented. The system is designed to implement the PES approach. This design suggests that the PES approach can also be implemented effectively and efficiently with moderate effort in a low-cost system.

Finally, simulation is done to compare the average performance of several PES scheduling algorithms. It is found that the "Longest Processing Time" scheduling has the best average performance among the algorithms being tested. Compared to a schedule without any reordering, the longest processing time schedule results in an average improvement of about 10% in completion time.

WU, SHYUE BIN. Interconnection Design and Resource Assignment for Large Multi-Microcomputer Systems, The Ohio State University, Autumn Quarter 1980.

Computer systems constructed by interconnecting a large number of microcomputers can offer substantial gains in performance, modularity, and robustness over conventional centralized systems. This dissertation addresses two issues in the design of large multi-microcomputer systems: how to design an interconnection structure in order to support interprocessor communication, and how to assign system resources in order to make a large number of microcomputers function together as a single integrated system.

A cluster structure is proposed as a conceptual scheme for interconnecting a large number of microcomputers. It is characterized by a set of structure parameters and a set of interconnection functions. Therefore, by specifying values for structure parameters and interconnection functions, one can specify a desired interconnection structure.

Performance analysis has been a key process in interconnection design. This dissertation proposes an analytical model to analyze message delay and traffic congestion problems. Different from others, this analysis considers the detailed interaction among system nodes. It shows the effect of structural topology on bus load and message delay.

Through the use of the cluster structure and the analytical model, this dissertation demonstrates how one can find an optimal structural topology so that more microcomputers can be interconnected and/or more message traffic can be supported without suffering serious system degradation. Case studies are presented to show how topological optimization can be done, subject to design constraints arising from system applications.

Minimum cost assignment of system resources is motivated by the desire to minimize system overhead, and is a key to the success for system integration. This dissertation describes how an efficient solution to the resource assignment problem can be obtained from using network flow algorithms. It describes how one can first partition a given module graph, representing software resources, and then map subsets of software resources to subsets of system nodes. Performance studies are also presented to show that, if a given graph is tree-like, our partition algorithm yields an optimal solution; otherwise, our partition algorithm yields a solution with near minimum cost.

This dissertation also discusses the effect of the resource assignment solution on system interconnection and system integration. It outlines an approach to applying the resource assignment solution to enhance system integration such as in control and scheduling problems.

RESEARCH AND DEVELOPMENT AWARDS

Equipment Grant

Title:	Equipment for the Laboratory for Database Systems Research
Principal Investigator:	David K. Hsiao
Investigator and Director:	Douglas S. Kerr
Sponsor:	External Research Program, Digital Equipment Corporation
Duration:	1980-82
Amount:	$222,155

Abstract:

A 50% discount of the market value of a multi-mini computer system consisting of one VAX 11/780 and two PDP-11/44s interconnected with parallel transfer buses has been granted to the Laboratory for Database Systems Research. The VAX 11/780 is configured with two 67-mbyte disks, one tape, four terminals, one console, one line printer and 1.5-mbyte primary memory. One PDP-11/44 has two 67-mbyte disks, one tape, two terminals and 256-kbyte primary memory, while the other PDP-11/44 has one 67-mbyte disk and 256-kbyte primary memory.

With matching funds from the Office of Naval Research, the following items have been installed in the Laboratory for the 1980-81 period. They are the two PDP-11/44s with 256-kbyte primary memory each, the line printer, the tape station, five CRT terminals, two PCLs, and three 67-mbyte disk drives. Plans are made to install the remaining equipment in the 1981-82 period.

Graduate Training Grant

Title:	Graduate Training Program in Biomedical Computing and Information Processing
Program Directors:	A. E. Petrarca, Associate Professor, Department of Computer and Information Science Gregory L. Trzebiatowski, Associate Dean, College of Medicine
Sponsor:	National Library of Medicine (NIH Grant LM 07023)
Duration:	7/1/80-6/30/82
Amount:	$148,249 (1980-81) $79,872 (1981-82)

Abstract:

In order to meet the needs for specialists in biomedical computing, an interdisciplinary Graduate Training Program in Biomedical Computing and Information Process-

ing was established through a joint effort of the School of Allied Medical Professions of the College of Medicine and the Department of Computer and Information Science. Students who are interested in the study and application of computer and information science to health care, medical education, and biomedical research may pursue, through one of the participating departments, the graduate degrees of Master of Science (via Allied Medical Professions of Computer and Information Science) and Doctor of Philosophy (via Computer and Information Science or appropriate Ph.D. granting departments in the College of Medicine).

Research Grants

Title:	Analyzing Program Methodologies Using Software Science
Principal Investigator:	Stuart H. Zweben
Sponsor:	U.S. Army Research Office (DAAG29-80-K-0061)
Duration:	8/1/80-7/31/83
Amount:	$146,579

Abstract: The area of applicability of various Software Science metrics to COBOL will be extended by conducting controlled experiments and by developing appropriate software tools to automate the analysis of data. The extent to which the metrics might be appropriate in evaluating the quality of computer software will also be investigated.

Title:	An Approach to Program Testing Based on Modularity
Principal Investigators:	Stuart H. Zweben Lee J. White
Sponsor:	National Science Foundation (MCS-8018769)
Duration:	1/15/81-1/31/83
Amount:	$120,120

Abstract: This project will investigate the extent to which testing of large computer programs can be facilitated by modularity in their development. Assuming that subunits (modules) of the total program are developed and tested independently, the idea is to make use of the information obtained in these unit tests when performing the validation of the large program. This would result in a significant reduction, over conventional methods, in the total amount of testing that would be required.

Title:	Computer Communication Protocols for Advanced Communication Systems
Principal Investigator:	Ming T. Liu
Sponsor:	U.S. Army Communications - Electronics Command Fort Monmouth (DAAK80-81-K-0104)
Duration:	6/1/81-12/31/81
Amount:	$25,000

Abstract: This project is concerned with the design of computer communication protocols for use in advanced communication systems. A formal model and software engineering techniques are applied to specify, verify, and implement multi-destination protocols for use in advanced computer-communication networks.

Title:	Extension and Application of a Theory of Information Flow and Analysis: The Use of Information in a Decision-Making Environment
Principal Investigator:	Clinton R. Foulk
Sponsor:	National Science Foundation (IST-7908327 A01)
Duration:	8/1/79-1/31/82
Amount:	$121,513

Abstract: This research program builds on previous research which has been underway at Ohio State University for the last few years. We now plan to extend our research in three different but complimentary directions by: 1) extending the basic theoretical work; 2) gathering additional data with the use of a flexible, sophisticated simulation model in order to establish new relationships and important parameters; and 3) developing, designing and carrying out experiments involving human subjects in order to obtain real data about use of information by decision-makers.

Title:	Knowledge Organization and Problem Solving for Diagnostic Tasks
Principal Investigator:	B. Chandrasekaran
Sponsor:	National Science Foundation (MCS-8103480)
Duration:	5/1/81-4/30/82
Amount:	$62,517

Abstract: An approach to knowledge organization and problem solving for expert systems which specialize in a certain class of tasks, viz, diagnosis, will be investigated. We propose that knowledge be decomposed into a collection of subspecialists who interact in specified ways to solve a diagnostic problem. We propose research on how the specialists should be coordinated and a high-level language that can be used to specify knowledge to the computer in different domains.

Title: Research in Database Computers and Systems

Principal Investigator: David K. Hsiao
Investigator: Douglas S. Kerr

Sponsor: Office of Naval Research (N00014-67-A-0232; N00014-75-C-0573)

Duration: 3/1/73-9/30/81

Amount: $856,495

Abstract: Present research focuses on the design and analysis of a multi-backend database system for high performance and great capacity. This system utilizes multiple and parallel mini-computer systems with identical software.

Of $217,790 funded for the 1980-81 period, the amount of $103,817 has been applied to the equipment purchase which is matched with an equal amount by an equipment grant from DEC. Consequently, mini-computers valued at $207,634 have been installed for the Database Computers and Systems Research.

Title: Statistical Methods for Algorithm Design and Analysis

Principal Investigator: Bruce W. Weide

Sponsor: National Science Foundation (MCS-7912688)

Duration: 10/1/79-9/30/81

Amount: $33,842

Abstract: Application of statistical methods at design time can lead to significant improvements in expected behavior of algorithms for discrete problems. For some problems, the use of sampling and density estimation, for example, leads to fast expected-time algorithms. Other problems, which cannot be solved exactly by any fast algorithms, are susceptible to probabilistic approximation algorithms which can also be designed and analyzed with the help of statistical methods.

Title:	Theoretical Foundations of Software Technology
Principal Investigators:	B. Chandrasekaran Lee J. White H. William Buttelmann
Sponsor:	U.S. Air Force Office of Scientific Research (F49620-79-C-0152)
Duration:	7/1/79-6/30/8[illegible]
Amount:	$381,212
Abstract:	This result will develop basic theoretical models and results in the areas of software and programming language structure and design, with the purpose of producing knowledge that will enable development of more reliable and transportable software. The current focus is on three areas: semi-automatic program testing, automatic program synthesis and computability.

Title:	Toward More Complicated Computer Imagery
Principal Investigator:	Franklin C. Crow
Sponsor:	National Science Foundation (MCS-7920977)
Duration:	1/15/80-6/30/81
Amount:	$79,342
Abstract:	Initial efforts will focus on the design of data structures to support efficient rendition of the same object at many different levels of detail. Subsequent work will focus on algorithms for the display of such objects and designed for distributed execution. Finally, the algorithms will be implemented and image sequences produced on a multicomputer facility.

APPENDIX A

CURRENT STATUS AND CAPSULE HISTORY OF
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

	SEPT '74	SEPT '75	SEPT '76	SEPT '77	SEPT '78	SEPT '79	SEPT '80	SEPT '81
A. Staff								
1. Full Time	20	21	22	20	21	27	24	28
2. Part Time	12	12	12	13	12	14	12	14
B. Graduate Students	198	201	182	197	198	200	200	200 (est)
C. Undergraduate Students	475	450	470	440	440	550	680	750 (est)
D. Course Enrollment (Autumn Quarter)	1925	2098	2290	2308	2568	2928	3100	3800 (est)

	'74-75	'75-76	'76-77	'77-78	'78-79	'79-80	'80-81
Students Taught	6876	7241	7615	7528	8447	9420	10,659
Baccalaureate Degrees Awarded	109	103	118	125	126	124	125
M.S. Degrees Awarded	58	64	70	54	59	56	80
Ph.D. Degrees Awarded	7	13	5	8	7	10	6
Ph.D. Degrees Awarded - Total	23	36	41	49	56	66	72
Applications for Graduate Study	355	325	333	335	479	509	525
Number of Graduate Students Supported	81	77	81	92	72	70	70

APPENDIX B

COMPUTER AND INFORMATION SCIENCE COURSE LISTING BY NUMBER AND TITLE

100 Computers in Society

201 Elementary Digital Computer Programming

211 Computer Programming for Problem Solving

212 Computer Data Processing

221 Programming and Algorithms I

222 Programming and Algorithms II

294 Group Studies

313 Introduction to File Design

321 Introduction to File Processing

380 File Design and Analysis

411 Design of On-Line Systems

489 Professional Practice in Industry

493 Individual Studies

505 Fundamental Concepts of Computer and Information Science

511 Computer Systems and Programming for Administrative Sciences

541 Survey of Numerical Methods

542 Introduction to Computing in the Humanities

543 Intermediate Digital Computer Programming

548 Computer Science for High School Teachers

551 Elements of Database Systems

555 Survey of Programming Languages

557 Minicomputer Programming Systems

560 Elements of Computer Systems Programming (Approved Spring 1981)

594 Group Studies

607 Mathematical Foundations of Computer and Information Science I

610 Principles of Man-Machine Interaction

640 Numerical Analysis

641 Computer Systems Programming I (Withdrawn Autumn 1980)

642 Numerical Linear Algebra

643 Linear Optimization Techniques in Information Processing

660 Introduction to Operating Systems

675 Introduction to Computer Architecture

676 Minicomputer and Microcomputer Systems

677 Computer Networks

680 Data Structures

693 Individual Studies

694 Group Studies

694L Biomedical Information Processing

6940 Introduction to Operating Systems (Number became 660, Autumn 1980)

694U Elements of Computer Systems Programming (Number became 560 Spring 1981)

707 Mathematical Foundations of Computer and Information Science II

712 Man-Machine Interface

720 Introduction to Linguistic Analysis

726 Theory of Finite Automata (Becomes Introduction to Automata and Language Theory, Spring 1982)

727 Turing Machines and Computability (Becomes Introduction to the Theory of Algorithms, Winter 1982)

728 Topics in Theory of Computing

730 Basic Concepts in Artificial Intelligence

735 Statistical Methods in Pattern Recognition

741 Comparative Operating Systems

745 Numerical Solution of Ordinary Differential Equations

746 Advanced Numerical Analysis

750 Modern Methods of Information Storage and Retrieval

751 Fundamentals of Document-Handling Information Systems

752 Techniques for Simulation of Information Systems

753 Theory of Indexing

755 Programming Languages

756 Compiler Design and Implementation

757 Software Engineering

760 Operating Systems

761 Introduction to Operating Systems: Laboratory

765 Management Information Systems

770 Database Systems (Effective Winter 1982)

775 Computer Architecture

780 File Structures (Becomes Analysis of Algorithms, Autumn 1981)

781 Aspects of Computer Graphics Systems

788 Intermediate Studies in Computer and Information Science

788.01 Theory of Information

788.02 Information Storage & Retrieval (Becomes Information Systems and Database Systems, Autumn 1981)

788.03 Theory of Automata

788.04 Artificial Intelligence

788.04A Topics in Artificial Intelligence

788.05 Pattern Recognition

788.06 Computer Systems Programming

788.06A Computer Center Organization and Management

788.06C Selected Topics in the Design & Implementation of Distributed Operating Systems

788.06D Data Models & Database Systems

788.07 Programming Languages

788.07B Selected Topics Related to the Design of Programming Environments

788.08 Computer Organization

788.09 Numerical Analysis

788.10 Man-Machine Interaction

788.10A Advanced Computer Graphics

788.11 Formal Languages

788.12 Management Information Systems

788.13 Biological Information Processing (Becomes Biomedical Information Systems, Autumn 1981)

788.14 Socio-Psychological Aspects of Information Processing (Becomes Computer Graphics, Autumn 1981)

793 Individual Studies

794 Group Studies

797 Interdepartmental Seminar

805 Information Theory in Physical Science

806 Cellular Automata and Models of Complex Systems

812 Computer and Information Science Research Methods

820 Computational Linguistics

835 Special Topics in Pattern Recognition

845 Numerical Solution of Partial Differential Equations

850 Theory of Information Retrieval I

852 Design and Analysis of Information Systems Simulations

855 Advanced Topics in Programming Languages

875 Advanced Computer Architecture

880 Advanced Theory of Computability

885 Seminar on Research Topics in Computer and Information Science

888 Advanced Studies in Computer and Information Science

888.01 Theory of Information

888.02 Information Storage & Retrieval (Becomes Information Systems and Database Systems, Autumn 1981)

888.03 Theory of Automata

888.03C Information & Coding for Efficiency, Reliability, and Security

888.04 Artificial Intelligence

888.04A Computational Linguistics

888.05 Pattern Recognition

888.06 Computer Systems Programming

888.06B Database Machines and Distributed Databases

888.07 Programming Languages

888.08 Computer Organization

888.09 Numerical Analysis

888.10 Man-Machine Interaction

888.11 Formal Languages

888.12 Management Information Systems

888.13 Biological Information Processing (Becomes Biomedical Information Systems, Autumn 1981)

888.14 Socio-Psychological Aspects of Information Processing (Becomes Computer Graphics, Autumn 1981)

889 Advanced Seminar in Computer and Information Science

899 Interdepartmental Seminar

999 Research

APPENDIX C

COMPUTER AND INFORMATION SCIENCE FACULTY

Professors

Lee J. White, Ph.D., (University of Michigan); Chairperson of the Department of Computer and Information Science; algorithm analysis and complexity, data structures, software engineering, program testing; joint appointment with Electrical Engineering.

Kenneth J. Breeding, Ph.D., (University of Illinois); computer organization and switching theory; joint appointment with Electrical Engineering.

Balakrishnan Chandrasekaran, Ph.D., (University of Pennsylvania); artificial intelligence, expert systems, knowledge-directed data bases, pattern recognition, computer program testing, interactive graphics.

Charles A. Csuri, M.A., (The Ohio State University); advancement of computer graphics technology in software and hardware (animation languages, data generation and real-time systems), use of computer technology in telecommunications; joint appointment with Art Education.

Tse-yun Feng, Ph.D., (University of Michigan); computer architecture, associative, parallel and concurrent processors/processing, processor/memory interconnection networks and communication processors.

Richard I. Hang, M.S., (The Ohio State University); computer graphics, engineering application of computers; joint appointment with Engineering Graphics.

David K. Hsiao, Ph.D., (University of Pennsylvania); systems programming, computer architecture, database management systems, access control and privacy protection of data, and database computers.

Clyde H. Kearns, M.S., (The Ohio State University); Professor Emeritus, Departments of Computer and Information Science and Engineering Graphics; computer graphics, engineering application of computers.

Robert D. LaRue, P.E., M.S., (University of Idaho); computer graphics, engineering application of computers; joint appointment with Engineering Graphics.

Ming-Tsan Liu, Ph.D., (University of Pennsylvania); computer architecture and organization, computer communications and networking; parallel and distributed processing; mini/micro computer systems, fault-tolerant computing systems.

Robert B. McGhee, Ph.D., (University of Southern California); robotics, switching theory, logical design; joint appointment with Electrical Engineering.

Roy F. Reeves, Ph.D., (Iowa State University); Professor Emeritus, Departments of Computer and Information Science, and Mathematics; numerical analysis, programming, and computer center management.

Jerome Rothstein, A.M., (Columbia University); information and entropy, foundations of physics, methodology, biocybernetics, automata theory, formal languages, cellular automata, parallel processing; joint appointment with Biophysics.

Charles Saltzer, Ph.D., (Brown University); coding theory, numerical analysis, automata theory; joint appointment with Mathematics.

Associate Professors

H. William Buttelmann, Ph.D., (University of North Carolina); microcomputers, small office systems, "friendly" systems, formal language theory, computational linguistics, language processing.

Ronald L. Ernst, Ph.D., (University of Wisconsin); man-computer interaction, decision systems, and general theory of human performance; joint appointment with Psychology.

Clinton R. Foulk, Ph.D., (University of Illinois); parallel processing, program analysis.

Douglas S. Kerr, Ph.D., (Purdue University); database systems, database machines, computer security and software engineering.

James C. Kinard, Ph.D., (Stanford University); accounting, management information systems, managerial decision making; joint appointment with Accounting.

Sandra A. Mamrak, Ph.D., (University of Illinois); distributed processing, operating systems, performance evaluation.

William F. Ogden, Ph.D., (Stanford University); software engineering, program verification, mathematical foundations of computing.

Anthony E. Petrarca, Ph.D., (University of New Hampshire); knowledge representation for information storage and retrieval, automatic indexing and classification, user interface, bio-medical information processing.

Stuart H. Zweben, Ph.D., (Purdue University); software engineering, programming methodology, analysis of algorithms, data structures.

Adjunct Associate Professors

James B. Randels, Ph.D., (The Ohio State University); Senior Programmer/Analyst, Instruction and Research Computer Center; computer operating systems and utilities, telecommunications applications, subroutine libraries, programming languages.

Lawrence L. Rose, Ph. D.;(Pennsylvania State University); Manager, Systems Simulation, Battelle Columbus Laboratories; discrete event simulation, software development, database systems.

James E. Rush, Ph.D., (University of Missouri); President of James E. Rush Associates, Inc., and President of Library Automation, Inc.; indexing theory, automated language processing, organization of information, parallel processing, structured programming, program testing, programming management, library automation and networking, documentation and standards.

Assistant Professors

Venkataraman Ashok, Ph.D., (Pennsylvania State University); analysis of algorithms and computational complexity; appointment Autumn 1981.

Bruce W. Ballard, Ph.D., (Duke University); programming languages, natural languages, and program synthesis.

Ramamoorthi Bhaskar, Ph.D., (Carnegie-Mellon University); accounting, artificial intelligence, cognitive psychology, managerial decision-making; joint appointment with Accounting.

Franklin C. Crow, Ph.D., (University of Utah); computer graphics, computer-aided design, multiprocessor and special purpose computer architecture.

John S. Gourlay, Ph.D., (University of Michigan); semantics of programming languages, analysis of parallelism, and the theory of testing; appointment Autumn 1981.

Dennis W. Leinbaugh, Ph.D., (University of Iowa); operating systems, hard-real-time, process synchronization, distributed operating systems, systems programming, computer architecture.

Timothy J. Long, Ph.D., (Purdue University); complexity theory, theory of computation, and algorithm analysis.

Sanjay Mittal, Ph.D., (The Ohio State University); artificial intelligence, knowledge-based systems, and data base modeling.

Kamesh Ramakrishna, Ph.D., (Carnegie-Mellon University); user-computer interaction, computational complexity, cognitive models of learning and instruction in complex task domains.

Jayashree Ramanathan, Ph.D., (Rice University); programming languages, computer systems, and software engineering.

Karstan Schwans, Ph.D., (Carnegie-Mellon University); distributed systems, programming languages, databases, systems modeling; appointment Autumn 1981.

Richard R. Underwood, Ph.D., (Stanford University); numerical linear algebra, solution of large sparse systems of equations, eigenvalue analysis, linear least squares problems, numerical solution of partial differential equations.

Bruce W. Weide, Ph.D., (Carnegie-Mellon University); analysis of algorithms, computational complexity, data structures, combinatorics, computer architecture, parallel and distributed computing, real-time programming.

Adjunct Assistant Professors

Bruce E. Flinchbaugh, Ph.D., (The Ohio State University); computational theory of vision, visual interpretation of motion and color, artificial intelligence; appointment Autumn 1981.

Lynn R. Ziegler, Ph.D., (University of Michigan); approximation theory, theoretical computer science, undergraduate curriculum; appointment Autumn 1981.

Instructor

Mary Beth Lohse, M.S., (University of Michigan); file processing, software engineering, and programming methodology; appointment Autumn 1981.

Visiting Faculty

Edna E. Cruz, M.E., (University of the Philippines); file processing, software engineering, and programming methodology; appointment Autumn 1981.

Neelamegam Soundararajan, Ph.D., (Bombay University); theory of computation, semantics of programming languages, semantics of parallel processing; appointment Autumn 1981.

Administrative and Professional Staff

Ernest Staveley, B.S., (U.S. Naval Post-Graduate School); Administrative Associate, and Assistant Director of CIS Research Center.

Celianna Taylor, B.S.L.S., (Graduate School of Library Science, Case-Western Reserve University); Senior Research Associate and Associate Professor of Library Administration; database design and development - home systems, public systems, and university dystems; library systems and management.

Faculty appointments, leaves of absence, and resignations

Venkataraman Ashok was appointed Assistant Professor of Computer and Information Science and comes from Pennsylvania State University where he recently received his Ph.D. in computer science. His areas of interest are analysis of algorithms and computational complexity.

Bruce W. Ballard resigned effective June 30, 1981. He will be an Assistant Professor in the Department of Computer Science, Duke University.

Edna E. Cruz was appointed Visiting Instructor of Computer and Information Science Autumn 1980. She received the Master of Engineering in Computer Science from the University of Philippines, April 1980. Her interests are file processing, software engineering, and programming methodology.

Ronald L. Ernst was granted an extension to his leave of absence. He will continue as Visiting Associate Professor in the Department of Computer Science, North Carolina State University for 1981-82.

Bruce E. Flinchbaugh was appointed Adjunct Assistant Professor. He received his Ph.D. degree from the Ohio State University in 1980. His areas of interest are computational theory of vision, visual interpretation of motion and color, and artificial intelligence, Appointment Autumn 1981.

John S. Gourlay was appointed Assistant Professor of Computer and Information Science and comes from the University of Michigan where he recently received the Ph.D. in computer and communication sciences. His research interests are semantics of programming languages, parallelism, and the theory of testing. Appointment Autumn 1981.

Clyde H. Kearns retired June 1981 and is now Professor Emeritus, Departments of Computer and Information Science, and Engineering Graphics.

Mary Beth Lohse was appointed Instructor beginning Autumn Quarter 1981. She received the M.S. degree from the University of Michigan. Her areas of interest are file processing, software engineering, and programming methodology.

Roy F. Reeves retired at the end of Summer quarter 1981. He was appointed Professor Emeritus, Departments of Computer and Information Science, and Mathematics.

Karstan Schwans was appointed Assistant Professor of Computer and Information Science and comes from Carnegie-Mellon University where he recently received his Ph.D. in computer science. His interests include semantics of programming languages, analysis of parallelism, and the theory of testing. Appointment, Autumn 1981.

Neelamegam Soundararajan returns to the department Autumn 1981 as Visiting Assistant Professor of Computer and Information Science. He comes from the Institute for Informatics, University of Oslo where he has completed a fellowship appointment. His Ph.D. was awarded from Bombay University, India. Research interests are theory of computation, semantics of programming languages, semantics of parallel processing.

Richard R. Underwood resigned at the end of Winter quarter 1981. He accepted a position with McDonnell Douglas Aircraft Company in St. Louis, Missouri.

APPENDIX D

COMPUTER AND INFORMATION SCIENCE SEMINAR SERIES

July 17, 1980 "Processing Inputs - Issues in Accessibility", Dr. Michael A. McAnulty, NTS Research Corporation.

July 28, 1980 "Interconnection Networks and Their Applications", Dr. Chuan-lin Wu, Assistant Professor, Computer Science, Wright State University.

October 2, 1980 "Computer Animation", Charles A. Csuri, Professor, Art Education and Computer and Information Science, The Ohio State University.

October 9, 1980 "Byte Wars -- The Computer Strikes Back", Franklin C. Crow, Assistant Professor, Department of Computer & Information Science, The Ohio State University.

October 16, 1980 "Real-Time Data-Flow Graphs: Programming and Implementation", Bruce W. Weide, Assistant Professor, Department of Computer & Information Science, The Ohio State University.

October 23, 1980 "Database Design and Conversion for Heterogeneous Databases", Dr. Randy H. Katz, Computer Corporation of America.

October 30, 1980 "INSYPS System: INtegrated SYstem for Process Studies", R. S. Ahluwalia, Assistant Professor, Industrial & System Engineering, The Ohio State University.

November 6, 1980 "Computer Go", David J. H. Brown, Visiting Assistant Professor, Department of Computer & Information Science, The Ohio State University.

November 13, 1980 "The Relative Neighborhood Graph, with an Application to Minimum Spanning Trees", Kenneth J. Supowit, Department of Computer Science, University of Illinois at Urbana- Champaign.

November 20, 1980 "An Examination of CODASYL Systems -- Status and Prospects", Eric K. Clemons, Assistant Professor, Department of Decision Sciences, The Wharton School, University of Pennsylvania.

November 25, 1980 "A General Theory of Automatic Program Synthesis", Carl H. Smith, Assistant Professor, Department of Computer Sciences, Purdue University.

December 1, 1980 "Fault-Tolerant Broadcast Problems", Arthur Liestman, Ph.D. Candidate, Computer Science Dept., University of Illinois.

January 8, 1981 "Analysis of a Class of Hybrid Page Replacement Policies", Ozalp Babaoglu, Ph.D. Candidate, Computer Science Department, University of California, Berkeley.

January 15, 1981 "Programming Languages for Bit-Serial Array Machines", Dennis M. Mancl, Ph. D. Candidate, Department of Computer Science, University of Illinois.

January 22, 1981 "Entity-Relationship Approach to Systems Analysis and Database Design", Dr. Peter P. Chen, Acting Associate Professor, UCLA Graduate School of Management.

January 29, 1981 "Specification and Synthesis of Synchronizers", Krithivasan Ramamritham, Ph.D. Candidate, Department of Computer Science, University of Utah.

January 29, 1981 "Interface Control for Centralized and Distributed Systems", Walter F. Tichy, Assistant Professor, Department of Computer Sciences, Purdue University.

February 5, 1981 "A Special Purpose Function Architecture for Some Relational Algebra Operations", Aral Ege, Ph. D. Candidate, Department of Industrial Engineering & Operations Research, Syracuse University.

February 12, 1981 "Artificial Intelligence in Medicine -- Accomplishments, Problems, and Prospects", Professor Saul Amarel, Chairman, Department of Computer Science, Rutgers University.

February 12, 1981 "Representing and Manipulating Inexact Information", Billy P. Buckles, Ph.D. Candidate, University of Alabama.

February 17, 1981 Methodology for High-Level Information System Design, Alexander T. Borgida, Assistant Professor, Department of Computer Science, University of Toronto.

February 18, 1981 "The Beta Spline: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures", Brian A. Barsky, Ph.D. Candidate, Department of Computer Science, University of Utah.

February 19, 1981 "Software for Multiple Processor Systems", Karsten Schwans, Ph.D. Candidate, Department of Computer Science, Carnegie-Mellon University.

February 26, 1981 "A Numerical Algorithm - An Extension of the Common Conjugate Gradient Method for the Solution of Sets of Linear Equations", Richard R. Underwood, Assistant Professor, Department of Computer and Information Science, The Ohio State University.

March 5, 1981 "File Allocation on Multiple Disk Systems", David H-C Du, Ph. D. Candidate, Computer Science Department, University of Washington.

March 10, 1981 "Algorithms for SIMD (Single-Instruction-Stream-Multiple-Data-Stream) Machines", Eliezer Dekel, Ph.D. Candidate, Department of Computer Science, University of Minnesota.

March 12, 1981 "An Incremental Family of Office Workstations", Robert Hudyma, Computer Systems Research Group, University of Toronto.

April 2, 1981 "Program Testing Based on Specifications", John S. Gourlay, Ph.D. Candidate, Computer and Communication Sciences, University of Michigan.

April 2, 1981 "Temporal Event Recognition: An Application to Left Ventricular Performance", John K. Tsotsos, Assistant Professor, Department of Computer Science, University of Toronto, and Canadian Heart Foundation Research Fellow.

April 9, 1981 "Computer Program Complexity Measures and Software Testing Methods", Sukhamay Kundu, Bell Laboratories, Murray Hill.

April 16, 1981 "Mechanisms for Process Management in Operating System Languages", Martin S. McKendry, Ph.D. Candidate, University of Illinois.

April 23, 1981 "Error Recovery in Concurrent Processes", Krishna Kant, University of Texas at Austin.

April 23, 1981 "Knowledge-Based Decision Support Systems in Medicine", James Reggia, MD, Assistant Professor, Department of Neurology, Doctoral Candidate, Department of Computer Science, University of Maryland.

April 28, 1981 "Design of a Multi-Language Editor", Mark R. Horton, Ph.D. Candidate, Computer Science Division, University of California, Berkeley.

April 29, 1981 "The Use of Requirements in Rigorous System Design", Deborah Baker, Ph.D. Candidate, Computer Science Department, University of Southern California.

May 14, 1981 "Performance Analysis of Broadcast Mode Communications with Acknowledgements Considerations", M. Y. "Medy" Elsanadidi, Ph.D. Candidate, Computer Science Department, UCLA.

May 21, 1981 "Possible Futures: A New Model of Concurrent Programs", William C. Rounds, Associate Professor, Computer and Communication Science, University of Michigan.

May 28, 1981 "Computing with Equation Schemata", Paul Chew, Ph.D. Candidate, Department of Computer Sciences, Purdue University.

APPENDIX E

PUBLICATIONS OF THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE STAFF

AMER, P. D.; MAMRAK, S. A. Statistical Procedures for Interactive Computer Computer Service Selection. In: Proceedings of the Fifth International Conference on Computer Communication, October 27-30, 1980, Atlanta, Georgia, pp. 695-702.

AMER, P. D.; MAMRAK, S. A. "Experimental Design for Comparing Interactive Computer Services. Computer Performance, Vol. 1, No. 3, December 1980, pp. 125-132.

CHANDRASEKARAN, B. Natural and Social System Metaphors for Distributed Problem Solving: Introduction to the Issue. In: IEEE Trans. Syst., Man & Cyber., January 1981.

CROW, F. C. "Three-Dimensional Computer Graphics." Byte Magazine, Part I, March 1981, Part II, April 1981. (Solicited)

FENG, T. Y.; WU, C. A Software Technique for Enhancing Performance of a Distributed Computer System. In: Proceedings COMPSAC '80, Chicago, October 29-31, 1980, pp. 274-280.

GOMEZ, F.; CHANDRASEKARAN, B. Knowledge Organizationaand Distribution for Medical Diagnosis. IEEE Trans. on Syst., Man & Cyber., January 1981.

HSIAO, D. K. Database Computers. In: Advances in Computers, Academic Press, 1980, Vol. 19.

HSIAO, D. K. Systems Programming - Concepts of Operating and Data Base Systems, Nippon Computer Kyokai, Tokyo, Japan, 1980, (in Japanese: a translation version of the same book is published by Addison-Wesley, U.S.A.).

HSIAO, D. K.; MENON J. The Impact of Auxiliary Information and Update Operations on Database Computer Architecture. In: Proceedings of International Congress on Applied Systems Research and Cybernetics, Pergamon Press, December 1980.

HSIAO, D. K. TODS - The First Three Years (1976-1978). In: ACM Transactions on Database Systems, 5, 4, December 1980, pp. 385-403.

LIU, M. T.; WU, S. B. A Partition Algorithm for Parallel and Distributed Processing. In: Proceedings of the 1980 International Conference on Parallel Processing, August 1980, pp. 254-255.

LIU, M. T.; WU, S. B. Assignment of Tasks and Resources for Distributed Processing. In: Proceedings of COMPCON Fall '80: Distributed Processing, September 1980, pp. 655-662.

LIU, M. T.; TSAY, D. P. Design of a Reconfigurable Front-end Processor for Computer Networks. In: Proceedings of the 10th International Symposium on Fault-Tolerant Computing, October 1-3, 1980, Kyoto, Japan, pp. 369-371.

LIU, M. T.; MAMRAK, S. A.; RAMANATHAN, J. The Distributed Double-Loop Computer Network (DDLCN). In: Proceedings of the 1980 ACM Annual Conference, October 27-29, 1980, Nashville, Tennessee, pp. 164-178.

LIU, M. T.; LI, C. M. Communicating Distributed Processes: A Language Concept for Distributed Programming in Distributed Database Systems. In: Proceedings of the Distributed Data Acquisition, Computing, and Control Symposium, December 3-5, 1980, Miami Beach, Florida, pp. 47-60.

LIU, M. T.; TSAY, D. P. Design of a Robust Network Front-end for the Distributed Double-Loop Computer Network (DDLCN). In: Proceedings of the Distributed Data Acquisition, Computing, and Control Symposium, December 3-5, 1980, Miami Beach, Florida, pp. 141-155.

LIU, M. T.; CHOU, C. P. A Concurrency Control Mechanism for a Partially Duplicated Distributed Database System. In: Proceedings of the 1980 Computer Networking Symposium, December 10, 1980, Gaithersburg, Maryland, pp. 26-34.

LIU, M. T.; LI, C. M. Minimum-Delay Process Communication: A Language Concept for Highly-Parallel Distributed Programming. In: Proceedings of the IEEE/CS COMPCON Spring '81, February 23-26, 1981, San Francisco, pp. 225-228.

LIU, M. T.; WU, S. B. A Cluster Structure as an Interconnection Network for Large Multimicrocomputer Systems. In: IEEE Transactions on Computers, April 1981, Vol. C-30, No. 4, pp. 254-264.

LIU, M. T.; LI, C. M. DISLANG: A Distributed Programming Language/System. In: Proceedings of the Second International Conference on Distributed Processing, April 8-10, 1981, Paris, France, pp. 162-172.

LIU, M. T.; TSAY, D. P.; CHOU, C. P. ; LI, C. M. "Design of the Distributed Double-Loop Computer Network (DDLCN)." Journal of Digital Systems, Volume 5, Nos. 1/2, Spring/Summer 1981, pp. 3-37.

LIU, M. T.; UMBAUGH, L. D. Adaptive Multi-Destination Protocols for Packet Radio Networks. In: Proceedings of the 1981 International Conference on Communications, June 1981, pp. 73.2.1-6.

RAMANATHAN, J.; ARTHUR, J. "Design of Analyzers for Selective Program Analysis." COMSAC, November 1980.

RAMANATHAN, J.; ARTHUR, J. Selective Analyzers for Programming Environments. In: IEEE Transactions on Software Engineering, January 1981.

RAMANATHAN, J.; SHUBRA, C. J. The Modeling of Problem Domains for Driving Program Development Systems. In: Proceedings of the Eighth Annual Symposium on the Principles of Programming Languages, January 1981.

RAMANATHAN, J.; KUO, J. "Concept Based Tool for Standardized Program Development." COMPSAC 1981.

ROTHSTEIN, J. Review of the Book, Electronic Imaging, edited by T. P. McLean and P. Schagen, Academic Press, London, 1979, appearing in Applied Optics, Vol. 19, No. 22, November 15, 1980, pp. 3774, 3781.

WEIDE, B. W. "Random Graphs and Graph Optimization Problems." SIAM Journal on Computing, August 1980, pp. 552-557.

WU, C.; FENG, T. Y. On A Class of Multistage Interconnection Networks. In: IEEE Transactions on Computers, Vol. C-29, No. 8, August 1980, pp. 694-702.

WU, C. L.; FENG, T. Y. On a Distributed-Processor Communication Architecture. In: Proceedings of the COMPCON '80 Fall Conference, September 24-26, 1980, Washington, D. C., pp. 599-605.

WU, C.; FENG, T. Y. The Reverse-Exchange Network. In: IEEE Transactions on Computers, Vol. C-29, No. 9, September 1980, pp. 801-811.

WU, C.; FENG, T. Y. Parallel Processing with a Modified Shuffle-Exchange Network. In: Proceedings of the ICCC '80 Conference, October 1-3, 1980, New York, pp. 390-392, 428.

ZEIL, S. J.; WHITE, L. J. Sufficient Test Sets for Path Analysis Testing Strategies. In: Proceedings of the 5th International Conference on Software Engineering, March 9-12, 1981, San Diego, California.

ZWEBEN, S. H.; BAKER, A. L. A Comparison of Measures on Control Flow Complexity. In: IEEE Transactions of Software Engineering, Vol. SE-6, No. 6, November 1980, pp. 506-512.

APPENDIX F

RECENT TECHNICAL REPORTS

1979

BANERJEE, J.; HSIAO, D. K. Parallel bitonic record sort- an effective algorithm for the realization of a post processor. March 1979. 22 pp. (OSU-CISRC-TR-79-1) (AD-A068 661/8GA).

BANERJEE, J.; HSIAO, D. K.; MENON, J. The clustering and security mechanisms of a database computer (DBC). April 1979. 112 pp. (OSU-CISRC-TR-79-2) (AD-A068 815/0GA)

DELUTIS, T. G.; CHANDLER, J. S. The Information Processing System Simulator (IPSS). "Language syntax semantics for the IPSS execution facility - Version 1" Volume I. 309 pp. (OSU-CISRC-TR-79-3).

DELUTIS, T. G.; CHANDLER, J. S.; BROWNSMITH, J. D.; WONG, P.; JOHNSTON, K. The Information Processing System Simulator (IPSS) "Language syntax and semantics for the IPSS Modeling Facility" Volume II. 1979. 325 pp. (OSU-CISRC-TR-79-4).

ROSE, L. L.; O'CONNOR, T. IDAS: Interactive design and analysis for simulation. 1979. 82 pp. (OSU-CISRC-TR-79-5).

HSIAO, D. K.; MENON, J. The post processing functions of a database computer. July 1979, 34 pp. (OSU-CISRC-TR-79-6).

CHANDLER, J. S. A multiple goal program model for the analysis of SSA district office service processing. August 1979. 33 pp. (OSU-CISRC-TR-79-7).

DELUTIS, T. G.; BROWNSMITH, J. D.; CHANDLER, J. S.; WONG, P.M.K. Methodologies for the performance evaluation of information processing systems. September 1979. 201 pp. (OSU-CISRC-TR-79-8).

1980

BLATTNER, M.; RAMANATHAN, J. TRIAD: A New Approach to Programming Methodology. January 1980, 48 pp. (OSU-CISRC-TR-80-1).

MAMRAK, S. A.; RAMANATHAN, J. A Programming/Operating System for a Distributed Computer System. February 1980. 28 pp. (OSU-CISRC-TR-80-2).

HSIAO, D. K.; MENON, J. Design and Analysis of Update Mechanisms of a Database Computer (DBC). June 1980. 127 pp. (OSU-CISRC-TR-80-3).

YOVITS, M. C.; FOULK, C. R.; ROSE, L. L. Information Flow and Analysis: Theory, Simulation, and Experiments. December 1979. 83 pp. (OSU-CISRC-TR-80-4).(PB80-180995).

FLINCHBAUGH, B. E.; CHANDRASEKARAN, B. A Theory of Spatio-Temporal Aggregation for Vision. April 1980. 43 pp. (OSU-CISRC-TR-80-5).

ZEIL, S. J.; WHITE, L. J. Sufficient Test Sets for Path Analysis Testing Strategies. July 1980, 29 pp. (OSU-CISRC-TR-80-6).

HSIAO, D. K.; MENON, M. J. Parallel Record-Sorting Methods for Hardware Realization. July 1980, 42 pp. (OSU-CISRC-TR-80-7) (AD/A 090 192).

HSIAO, D. K.; MENON, M. J. Design and Analysis of Relational Join Operations of a Database Computer (DBC). September 1980, 92 pp. (OSU-CISRC-TR-80-8).

1981

HSIAO, D. K.; OZSU, T. M. A Survey of Concurrency Control Mechanisms for Centralized and Distributed Databases. February 1981, 82 pp. (OSU-CISRC-TR-81-1).

MAMRAK, S. A.; BERK, T. S. The DESPERANTO Research Project. February 1981, 47 pp. (OSU-CISRC-TR-81-2).

LEINBAUGH, D. S. High Level Specification and Implementation of Resource Sharing. February 1981. 25 pp. (OSU-CISRC-TR-81-3).

BALLARD, B. W. A Methodology for Evaluating Near-Prototype Natural Language Processors. May 1981, 43 pp. (OSU-CISRC-TR-81-4).

HALEY, A.; ZWEBEN, S. An Approach to Reliable Integration Testing. May 1981, 41 pp. (OSU-CISRC-TR-81-5).

AGUILAR, L. Economic Broadcast Acknowledgement for Store-and-Forward Packet Switching. June 1981, 43 pp. (OSU-CISRC-TR-81-6).

HSIAO, D. K.; MENON, M. J. Design and Analysis of a Multi-Backend Database System for Performance Improvement, Functionality Expansion and Capacity Growth (Part I). July 1981, 186 pp. (OSU-CISRC-TR-81-7).

HSIAO, D. K.; MENON, M. J. Design and Analysis of a Multi-Backend Database System for Performance Improvement, Functionality Expansion and Capacity Growth (Part II). August 1981, 117 pp. (OSU-CISRC-TR-81-8).

APPENDIX G

ACTIVITIES OF THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE STAFF

M. E. Brown presented "Preliminary Design of a Highly Parallel Architecture for Real-Time Applications" at the 18th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois, October 8, 1980. Co-author was B. W. Weide.

B. Chandrasekaran presented an invited paper entitled "From Percept to Concept and Back Again" at the American Association for Advancement of Science's Symposium on 'Mechanical Intelligence and Perception: Premises & Prospects'", Toronto, Ontario, Canada, January 4, 1981. Co-author was B. E. Flinchbaugh.

B. Chandrasekaran presented an invited talk on "Knowledge Organization and Distribution for Diagnostic Tasks" at the Artificial Intelligence Seminar Series of the Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, January 5, 1981.

T. Feng organized the 1980 International Conference on Parallel Processing, August 26-29, 1980, Harbor Springs, Michigan. The Conference was jointly sponsored by the IEEE Computer Society and The Department of Computer and Information Science, The Ohio State University.

T. Feng led the IEEE Study Group on Computers on a visit to The People's Republic of China at the invitation of the Chinese Institute of Electronics, September 17 - October 4, 1980. He is presently the President of the IEEE Computer Society.

T. Feng presented "A Software Technique for Enhancing Performance of a Distributed Computer System" at COMPSAC '80, Chicago, October 29-31, 1980. The paper was also published in the conference proceedings, pp. 274-280. Co-author was C.L. Wu.

T. Feng cut the ribbon marking the Grand Opening of the IEEE Computer Society's new West Coast Office in Los Alamitos, California, on February 26, 1981. The building was purchased and remodeled while Dr. Feng was president of the Society. At the ceremony, Dr. Feng was presented with a Special Award for his "distinguished service as President of IEEE Computer Society during the years 1979-1980".

B. E. Flinchbaugh presented an invited talk on "A Computational Theory of Spatio-Temporal Aggregation for Vision" to the Computer Science Department, University of Toronto, Toronto, Ontario, Canada, January 26, 1981.

D. J. Hogan presented "External Sorting Revisited: Application of Distributive Methods" at the 18th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois, October 8, 1980. Co-author was B. W. Weide.

D. K. Hsiao served as an External Examiner for the National Bureau of Standards on Data Model Processor Research review, Sept. 11-12, 1980.

D. K. Hsiao was elected by the members of the Association for Computing Machinery (ACM) as the Member-at-Large for the ACM Council, effective October 1980 - September 1984.

D. K. Hsiao gave the opening speech with Dr. William Armstrong (of the University of Montreal) at the 1980 International Conference on Very Large Databases, Montreal, Canada, October 1, 1980.

D. K. Hsiao gave presentations on Database Computers to the following:

Bell Laboratories, Holmdel, NJ, September 15-16, 1980.

INFOTECH Seminar on Database, London, England, October 15, 1980.

National Security Agency, Baltimore, MD, October 22, 1980.

D. K. Hsiao gave a presentation on Data Base Education at COMPCON, Washington, DC, September 26, 1980.

D. K. Hsiao, as a speaker for the IEEE Computer Society's Distinguished Speakers Series, presented talks on "Database Computers" to

IEEE Computer Society's student chapters at the University of South Florida (Tampa) on November 25, 1980.

Texas A&M University (College Station) on December 3, 1980.

IEEE Computer Society's local chapters in College Station, Texas on December 3, and in Houston, Texas on December 4, 1980.

D. K. Hsiao gave a three-day seminar on "Computer Security" in Quito, Ecuador, November 27-29, 1980 and a one-day seminar on the same topic in Bogota, Colombia on December 1, 1980.

D. K. Hsiao presented the OSU database computer work at the International Congress on Applied Systems Research and Cybernetics in Acapulco, Mexico, on December 15, 1980. Co-author was J. Menon.

D. K. Hsiao presented an invited talk on "Database Computers" to the IBM Scientific Center's Colloquium in Mexico City, December 11, 1980.

D. K. Hsiao served as a reviewer on a Review Panel for the National Science Foundation's Computer Equipment Funds, Washington, D. C., January 12-13, 1981.

M. T. Liu presented "Design of a Reconfigurable Front-end Processor for Computer Networks" at the 10th International Symposium on Fault-Tolerant Computing, October 1-3, 1980, Kyoto, Japan. The paper appeared in the Conference Proceedings, pp. 369-71. Co-author was D. P. Tsay.

M. T. Liu presented "The Distributed Double-Loop Computer Network (DDLCN)" at the 1980 ACM Annual Conference, October 27-29, 1980, Nashville, Tennessee. The paper appeared in the Conference Proceedings, pp. 164-178. Co-authors were S. A. Mamrak and J. Ramanathan.

M. Liu presented "Communicating Distributed Processes: A Language Concept for Distributed Programming in Distributed Database Systems" and "Design of a Robust Network Front-End for the Distributed Double-Loop Computer Network (DDLCN)" at the Distributed Data Acquisition, Computing, and Control Symposium, Miami Beach, Florida, December 3-5, 1980. Co-authors were C. M. Li and D. P. Tsay, respectively. The papers appeared in the Conference Proceedings, pp. 47-60 and pp. 141-155, respectively.

M. T. Liu presented "A Concurrency Control Mechanism for a Partially Duplicated Distributed Database System" at the 1980 Computer Networking Symposium, Gaithersburg, Maryland, December 10, 1980. Co-author was C. P. Chou. The paper appeared in the Conference Proceedings, pp. 26-34.

M. T. Liu presented "Minimum-Delay Process Communication: A Language Concept for Highly-Parallel Distributed Programming" at IEEE/CS COMPCON Spring '81, February 23-26, 1981, San Francisco. The paper was published in the Conference Proceedings, pp. 225-228. Co-author was C. M. Li.

M. T. Liu presented "On Local Networking and Distributed Processing" at Wright State University, Dayton, Ohio, April 2, 1981.

M. T. Liu presented "On Distributed Processing and Local Networking" at North Carolina State University, Raleigh, NC, May 7, 1981.

M. T. Liu and J. Rothstein have been invited to be Distinguished Visitors of the IEEE Computer Society for 1981-82; they have been appointed Guest Editors of a special issue of the IEEE Transactions on Computers for parallel and distributed processing, scheduled for publication in December 1982; and they are serving as program co-chairmen for the 1981 International Conference on Parallel Processing, Bellaire, Michigan, August 25-28, 1981.

J. Rothstein presented "The Physics of Selective Systems: Computers and Biology" in a Biophysics Seminar, The Ohio State University, May 11, 1981.

L. J. White participated in a panel discussion entitled "Computer Science: Problems for Mathematics Departments at Small Colleges and Universities" at a meeting of the Ohio Section of the Mathematical Association of America held at John Carroll University, Cleveland, Ohio, on October 7, 1980.

L. J. White served on a review team consisting of six academic and industrial researchers to evaluate the Ph.D. programs in computer science in the Texas system of Colleges and Universities, February 2-5, 1981. Proposals for new Ph.D. programs in computer science were evaluated at the University of Houston and a joint program at North Texas State University, East Texas State University, and Texas Woman's University. Existing Ph.D. programs at Texas A & M, University of Texas at Dallas, University of Texas at Arlington, University of Texas Health Science Center at Dallas, and University of Texas at Austin were reviewed and evaluated.

L. J. White presented an invited talk titled "An Overview of Research Activities in Software Engineering at Ohio State University" to the TRW Corporation, Redondo Beach, CA, March 20, 1981.

C. L. Wu presented "On a Distributed-Processor Communication Architecture" at COMPCON '80 Fall, Washington, D. C., Sept. 24-26, 1980 . The paper was also published in the conference proceedings, pp. 599-605. Co-author was T. Y. Feng .

C. L. Wu presented "Parallel Processing with a Modified Shuffle-Exchange Network" at ICCC '80, New York, October 1-3, 1980 . The paper was also published in the conference proceedings, pp. 390-392 and p. 428. Co-author was T. Y. Feng.

S. J. Zeil and L. J. White presented a paper entitled "Sufficient Test Sets for Path Analysis Testing Strategies" at the 5th International Conference on Software Engineering, San Diego, CA, March 9-12, 1981.

S. H. Zweben presented "An Approach to Computer Program Testing" to the Baylor University ACM Chapter, Waco, Texas, December 9, 1980 and to the Dallas Chapter of the ACM, Dallas, Texas, December 10, 1981.

S. H. Zweben presented "A Tutorial on Software Science" to the Association for Computing Machinery's (ACM) Niagara Frontier Chapter, Buffalo, NY, March 17, 1981, and to ACM's Rochester Chapter, Rochester, NY, March 18, 1981.

S. H. Zweben presented "An Approach to Computer Program Testing" to the ACM's Alberta Chapter, Calgary and Edmonton, Alberta, March 23, 1981, and to ACM's Madison Chapter, Madison, WI, March 25, 1981 .

APPENDIX H

DOCTORATES AWARDED

1971-72

CAMERON, JAMES S. Automatic Document Pseudoclassification and Retrival by Word Frequency Techniques

EKONG, VICTOR J. Rate of Convergence of Hermite Interpolation Based on the Roots of Certain Jacobi Polynomials

GORDON, ROBERT The Organization and Control of a Slave Memory Hierarchy

LANDRY, B. CLOVIS A Theory of Indexing: Indexing Theory as a Model for Information Storage and Retrieval

1972-73

DEFANTI, THOMAS A. The Graphics Symbiosis System - an Interactive Mini-Computer Animation Graphics Language Designed for Habitability and Extensibility

GELPERIN, DAVID H. Clause Deletion in Resolution Theorem Proving

HARRIS, DAVID R. GOLDA: A Graphical On-Line System for Data Analysis

LAY, W. MICHAEL The Double-KWIC Coordinate Indexing Technique: Theory, Design, and Implementation

MATHIS, BETTY ANN Techniques for the Evaluation and Improvement of Computer-Produced Abstracts

WEIMAN, CARL F. R. Pattern Recognition by Retina-Like Devices

WHITTEMORE, BRUCE J. A Generalized Decision Model for the Analysis of Information

YOUNG, CAROL E. Development of Language Analysis Procedures with Application to Automatic Indexing

1973-74

CHAN, PAUL SUI-YUEN An Investigation of Symmetric Radix for Computer Arithmetic

GILLENSON, MARK L. The Interactive Generation of Facial Images on a CRT Using a Heuristic Strategy

HEPLER, STEPHEN PHILIP Use of Probabilistic Automata as Models of Human Performance

WANG, PAUL TIING RENN Bandwidth Minimization, Reducibility Decomposition, and Triangulation of Sparse Matrices

1974-75

BEUG, JAMES L. Human Extrapolation of Strings Generated by Ordered Cyclic Finite State Grammars

DOHERTY, MICHAEL E. A Heuristic for Minimum Set Covers Using Plausability Ordered Searches

FOURNIER, SERGE The Architecture of a Grammar-Programmable High-Level Language Machine

LONGE, OLUWUMI An Index of Smoothness for Computer Program Flowgraphs

MCCAULEY, EDWIN JOHN A Model for Data Secure Systems

PETRY, FREDERICK E. Program Inference from Example Computations Represented by Memory Snapshot Traces

SU, HUI-YANG Pagination of Programs for Virtual Memory Systems

1975-76

BAUM, RICHARD I. The Architectural Design of a Secure Data Base Management System

DASARATHY, BALAKRISHNAN Some Maximum, Location and Pattern Separation Problems: Theory and Algorithms

HARTSON, H. REX Languages for Specifying Protection Requirements in Data Base Systems - A Semantic Model

JUELICH, OTTO C. Compilation of Sequential Programs for Parallel Execution

KALMEY, DONALD L. Comparative Studies Towards the Performance Evaluation of Software for Solving Systems for Nonlinear Equations

KAR, GAUTAM A Distance Measure for Automatic Sequential Document Classification System

MOSHELL, JACK MICHAEL Parallel Recognition of Formal Languages by Cellular Automata

MUFTIC, SEAD Design and Operations of a Secure Computer System

PYSTER, ARTHUR B. Formal Translation of Phrase-Structured Languages

REAMES, CECIL C. System Design of the Distributed Loop Computer Network

RUSSO, PHILLIP M. Cellular Networks and Algorithms for Parallel Processing of Non-Numeric Data Encountered in Information Storage and Retrieval Applications

SANTHANAM, VISWANATHAN Prefix Encoding with Arbitrary Cost Code Symbols

SRIHARI, SARGUR N. Comparative Evaluation of Stored-Pattern Classifiers for Radar Aircraft Identification

1976-77

CHENG, TU-TING Design Consideration for Distributed Data Bases in Computer Networks

GUDES, EHUD An Application of Cryptography to Data Base Security

ISAACS, DOV Computer Operating System Facilities for the Automatic Control and Activity Scheduling of Computer-Based Management Systems

KRISHNASWAMY, RAMACHANDRAN Methodology and Generation of Language Translators

LEGGETT, ERNEST W., JR. Tools and Techniques for Classifying NP-Hard Problems

1977-78

BABIC, GOJKO Performance Analysis of the Distributed Loop Computer Network

CHANDLER, JOHN S. A Multi-Stage Multi-Criteria Approach to Information System Design

COHEN, DAVID Design of Event Driven Protection Mechanisms

COHEN, EDWARD I. A Finite Domain-Testing Strategy for Computer Program Testing

KANNON, KRISHNAMURTHI The Design and Performance of a Database Computer

LAKSHMANAN, K. B. Decision Making with Finite Memory Devices

MARIK, DELORES A. Grammatical Inference of Regular and Context- Free Language

PARENT, RICHARD E. Computer Graphics Sculptors' Studio - An Approach to Three-Dimensional Data Generation

1978-79

AMER, PAUL D. Experimental Design for Computer Comparison and Selection

BANERJEE, JAYANTA Performance Analysis and Design Methodology for Implementing Database Systems on New Database Machines

BROWNSMITH, JOSEPH D. A Methodology for the Performance Evaluation of Database Systems

DICKEY, FREDERICK J. Translations Between Programming Languages

LEE, MARY JANE An Analysis and Evaluation of Structure Decision Systems

NATARAJAN, K. S. A Graph-Theoretic Approach to Optimal File Allocation in Distributed Computer Networks

WANG, JIN-TUU Design of a Mixed Voice/Data Transmission System for Computer Communication

1979-80

BAKER, ALBERT L. Software Science and Program Complexity Measures

FLINCHBAUGH, BRUCE E. A Computational Theory of Spatio-Temporal Aggregation for Visual Analysis of Objects in Dynamic Environments

JAPPINEN, HARRY A Perception-Based Developmental Skill Acquisition System

KO, KER-I Computational Complexity of Real Functions and Polynomial Time Approximation

KWASNY, STAN C. Treatment of Ungrammatical and Extra-Grammatical Phenomena in Natural Language Understanding Systems

MELLBY, JOHN ROLF The Recognition of Straight Line Patterns by Bus Automatons Using Parallel Processing

PARDO, ROBERTO Interprocess Communication and Synchronization for Distributed Systems

TENG, ALBERT Y. Protocol Constructions for Communication Networks

WOLF, JACOB J., III. Design and Analysis of the Distributed Double-Loop Computer Network (DDLCN)

WONG, PATRICK M. K. A Methodology for the Definition of Data Base Workloads: An Extension to the IPSS Methodology

1980-81

CHOU, CHUEN-PU System Design of the Distributed Loop Database System (DLDBS)

LI, CHUNG-MING Communicating Distributed Processes: A Programming Language Concept for Distributed Systems

MITTAL, SANJAY Design of a Distributed Medical Diagnosis and Data Base System

TSAY, DUEN-PING MIKE: A Network Operating System for the Distributed Double-Loop Computer Network

WANG, PONG-SHENG Computer Architecture for Parallel Execution of High-Level Language Programs

WU, SHYUE BIN Interconnection Design and Resource Assignment for Large Multi-Microcomputer Systems

APPENDIX I

STUDENTS IN THE FINAL STAGES OF RESEARCH LEADING TO THE PH.D. DEGREE

Name	Dissertation Topic/Title	Advisor
Aguilar, Lorenzo	Multi-cast Computer Communication Services	Weide, B.
Aitken, Jan A.	A Methodology for the Evaluation of Languages for the Development of Computer-Based Information System Performance Models (Ph.D. awarded Summer 1981)	Kerr, D.
Ayen, William E.	Computer systems Programming/Performance Evaluation	Mamrak, S.
Brinkman, Barry J.	The Development and Evaluation of Interactive Aids for Search Profile Construction in Document Retrieval Systems	Petrarca, A.
Carlson, Wayne E.	Computer Graphics Research	Crow, F.
Champion, David M.	Problems in Associative Memory Systems	Rothstein, J.
Davis, L. Anne	Recognition of Conics by Bus Automata	Rothstein, J.
Fried, John B.	Development and Evaluation of the Keyword Decision Tree (KDT) Classification	Petrarca, A.
Gomez, Fernando J.	On General and Expert Knowledge-Based Methods in Problem Solving (Ph.D. awarded Summer 1981)	Chandrasekaran, B.
Gunji, Takao	Toward a Computational Theory of Pragmatics -- Extensions of Montagu Grammar	Buttelmann, H.W.
Haley, Allen W.	Reliable Module Integration Testing	Zweben, S.
Hall, William E.	Programming Systems	Buttelmann, H.W.
Hochstettler, William H.	A Methodology for the Specification of Information Systems to Support Macro Estimating and Project Management	Ernst, R.
Juell, Paul L.	Improvements in the Style of Computer Generated Natural Language Text (Ph.D. awarded Summer 1981)	Buttelmann, H.W.
Kuo, Hong-Chih Jeremy	Customizable Editor	Ramanathan, J.

Name	Dissertation Topic/Title	Advisor
Lin, Jy-Jine	Computer Architecture for Very Large Online Distributed Database Systems	Liu, M.
Menon, Jaishankar M.	Design and Analysis of a Multi-Backend Database System for Performance Improvement and Capacity Growth	Hsiao, D.
Ozsu, Tamer M.	Research in Database Machines	Hsiao, D.
Perry, Doyt L.	Computability and Complexity Issues in Translator Generation	Buttelmann, H.W.
Shubra, Charles	Modeling of the File Processing Domain	Ramanathan, J.
Soni, Dilip A.	A Model for a Customizable Workstation Environment	Ramanathan, J.
Stalcup, William S.	Techniques for the Evaluation and Improvement of Automatic Vocabulary Control in Printed Indexes	Petrarca,A.
Zeil, Steven J.	Selecting Sufficient Sets of Test Paths for Program Testing (Ph.D. awarded Summer 1981)	White, L.